MASTER EN SCIENCES INFORMATIQUES

Réflexion sur le modèle TCP/IP
comparaison et analyse d'implémentation

Daisomont, Jean-Paul

Award date:
1988

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

REFLEXION SUR LE MODELE TCP/IP
COMPARAISON ET ANALYSE D'IMPLEMENTATION

Jean-Paul Daisomont

Mémoire présenté pour l'obtention
du grade de
Licencié et Maître en informatique
par

Jean-Paul Daisomont
1987-1988

Promoteur :
Prof. Ph. van Bastelaer

F.U.N.D.P. Namur

Année académique 1987-1988

Faculté : Institut d'informatique
Adresse : Rue Grandgagnage, 21
B-5000 Namur
081-22 90 65

Nom et prénom de l'étudiant : Jean-Paul Daisomont

Titre du mémoire :

REFLEXION SUR LE MODELE TCP/IP
comparaison et analyse d'implémentation

Résumé

Le modèle TCP/IP a été conçu pour connecter différents types de support de communication utilisant la technique du paquet. Ces supports sont reliés entre eux par des passerelles. Ensemble, ils forment un catenet. Le modèle TCP/IP fournit un service fiable de communication entre processus s'exécutant sur des ordinateurs hôtes attachés au catenet. Le coeur de ce modèle est formé par le protocole IP (Internet Protocol) et par le protocole TCP (Transmission Control Protocol). Le protocole IP fournit un service de communication non fiable entre ordinateurs du catenet. Il correspond assez bien au protocole ISO 8473 de l'architecture OSI. Le protocole TCP rend fiable ce service de communication. Il est proche du protocole de classe 4 de la couche transport dans l'architecture OSI.

Dans ce travail, nous présentons d'abord la démarche suivie par le DoD et l'ISO lors du développement des architectures DoD et OSI. Ensuite, nous décrivons les protocoles IP, ARP (Address Resolution Protocol), RIP (Routing Information Protocol), ICMP (Internet Control Message Protocol) et TCP. Enfin, nous présentons les interfaces entre les différents modules associés aux protocoles IP et TCP pour la version 4.2BSD du modèle TCP/IP.

Abstract

The TCP/IP model was developed to connect a variety of transmission media based on packet-technology. These transmission media are connected together with gateways to form a catenet. The TCP/IP model provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. The heart of the TCP/IP model is formed by the IP protocol and by the TCP protocol. The IP (Internet Protocol) protocol is not designed to be absolutely reliable. On the other hand, the TCP (Transmission Control Protocol) protocol is intended for use as a highly reliable host-to-host protocol. The IP protocol corresponds fairly to the ISO 8473 protocol of the OSI Reference Model and the TCP protocol to the class 4 protocol of the transport layer defined by the ISO.

In this work we first present the development process for the OSI model and the TCP/IP model. After that we describe the protocols IP, ARP (Address Resolution Protocol), RIP (Routing Information Protocol), ICMP (Internet Control Message Protocol) and TCP. Finally we show the interfaces between the modules associated with the protocols IP and TCP for the version 4.2BSD of the TCP/IP model.

A cet endroit, je voudrais adresser un mot de remerciement à toutes les personnes qui ont contribué à la rédaction de ce mémoire.

Tout d'abord je voudrais remercier mon promoteur, le Professeur Ph. van Bastelaer, pour le suivi de ce travail. Ses commentaires et ses questions critiques m'ont été très utiles.

Je suis également reconnaissant à Messieurs les assistants P. Geurts, S. Simonet et B. Delcourt pour leur disponibilité et leurs explications concernant le système UNIX et les architectures DOD et ISO.

Enfin, je tiens à marquer ma gratitude à Monsieur P.A. Godelaine pour les nombreuses explications concernant les architectures ISO et XNS.

LISTE DES ABREVIATIONS

CCITT	:	Comité Consultatif International de Télégraphie et de Téléphone.
CEN	:	Comité Européen de Normalisation.
CENELEN	:	Comité Européen de Normalisation pour l'ELECtricité.
CEPT	:	Conférence Européenne des Postes et Télécommunications.
CERN	:	Centre Européen de Recherche Nucléaire.
DNA	:	Digital Network Architecture.
DOD	:	Department Of Defense.
ECMA	:	European Computer Manufacturers Association.
ETCD	:	Equipement de Terminaison de Circuit de Données.
ETTD	:	Equipement Terminal de Traitement de Données.
ICMP	:	Internet Control Message Protocol.
IEEE	:	Institute of Electrical and Electronic Engineers.
IP	:	Internet Protocol.
ISO	:	International Standards Organization.
LLC	:	Logical Link Control.
MAC	:	Meduim Access Control.
MAP	:	Manufacturing Automation Protocol.
PUP	:	Parc Universal Protocol.
RFC	:	Request For Comments.
SNA	:	Systems Network Architecture.
TCP	:	Transmission Control Protocol.
TOP	:	Technical and Office Protocol.
UDP	:	User Datagram Protocol.
XNS	:	Xerox Network System.

PLAN DU MEMOIRE

INTRODUCTION

1. Exposé de l'objet du mémoire

L'objet de ce mémoire est avant tout de décrire la normalisation qui a été définie aux Etats-Unis par le Département de la Défense. Cette normalisation est généralement appelée normalisation DoD (Department of Defense) ou modèle DoD. Elle cherche à gérer l'échange d'informations entre ordinateurs ouverts à la communication avec d'autres ordinateurs. Le plus souvent, ces échanges sont rendus possibles parce que ces ordinateurs sont connectés à des réseaux de communication.

La normalisation DoD n'est pas la seule normalisation qui ait été définie dans ce domaine. Depuis les années septante, de nombreux travaux de spécification ont été entrepris

- par des organisations de normalisation (ISO, IEEE, CCITT, ECMA),
- par des organismes universitaires ou de recherche (CERN),
- par des organisations gouvernementales (DoD), mais également
- par des constructeurs (IBM, Digital, ...).

La normalisation la plus connue est, sans conteste, le Modèle de Référence OSI (Open Systems Interconnection) proposé par l'Organisation Internationale de Standardisation (ISO, pour International Standards Organization). Ce modèle est accepté mondialement comme référence. Par rapport à lui, le modèle DoD présente l'avantage d'avoir fait l'objet d'un grand nombre de recherches et d'expériences. La comparaison de ces deux modèles pourra donc s'avérer très intéressante. Cette comparaison sera le deuxième objet de ce mémoire.

L'implémentation la plus répandue du modèle DoD est celle qui est fournie avec la version 4.2BSD de UNIX. Cette version est disponible à l'Institut d'Informatique. Nous avons donc profité de cette opportunité pour la décrire. Ceci constitue le troisième objet de ce mémoire.

2. Situation du problème

Après une période où l'informatique a été concentrée dans de grands centres de calcul, il est apparu depuis une dizaine d'années un mouvement de décentralisation. Cette évolution a été rendue possible par le développement quasi-simultané des mini-ordinateurs, des micro-ordinateurs et des réseaux de communication.

Pour gérer l'échange de messages entre différents ordinateurs, de nombreux travaux de normalisation ont été entrepris. Le modèle DoD ainsi que le modèle OSI sont les deux plus connus entre eux.

Pour permettre l'échange de messages entre ordinateurs connectés à des réseaux éventuellement distincts mais interconnectés, la normalisation DoD a défini une architecture en quatre niveaux. Cette architecture s'appuie sur deux protocoles très importants . Il s'agit du protocole IP (Internet Protocol) et du protocole TCP (Transmission Control Protocol). Ils forment le coeur du modèle DoD. C'est pourquoi ce modèle est aussi appelé normalisation TCP/IP. Le protocole IP fournit un service de transport de données peu fiable entre ordinateurs connectés à des réseaux éventuellement différents mais reliés au moyen de passerelles. Dans la normalisation DoD, ces ordinateurs sont appelés ordinateurs hôtes pour signaler qu'ils sont ouverts sur l'extérieur. Pour rendre l'échange des données fiable, le DoD a défini le protocole TCP. Signalons encore que le modèle DoD a ajouté à ces deux protocoles un troisième protocole qui permet à un ordinateur hôte d'obtenir des renseignements sur l'état de fonctionnement des réseaux. Ce protocole est appelé protocole ICMP (Internet Control Message Protocol).

Le modèle OSI, quant à lui, est constitué de sept couches. La couche qui est responsable de l'échange des messages entre ordinateurs s'appelle couche réseau. Récemment, de nombreux projets de norme relatifs à cette couche ont été élaborés. Ils concernent, entre autres, l'organisation interne de la couche réseau (ISO/DIS 8648), le problème de l'interconnexion des réseaux (ISO/DIS 8473) et la définition d'une adresse OSI pour cette couche (ISO/DIS 8348/ADD2).

Dans ce mémoire, nous avons essayé chaque fois que cela était possible, de compléter les choix opérés dans le modèle DoD par les derniers développements enregistrés dans le cadre du modèle OSI. Ceci s'applique en particulier, aux trois projets de norme que nous venons de citer.

Les problèmes liés à la normalisation dans le domaine de la communication ont fait l'objet de nombreux ouvrages. Pour ce travail, nous avons surtout consulté les références suivantes : (Macchi 87), (Pujolle 87), (Stallings 85) et (Tanenbaum 81). Les deux premiers ouvrages ont été rédigés en français tandis que les deux suivants l'ont été en anglais. Pour ce qui est de la normalisation OSI, il est sorti en 1987 un livre qui fait remarquablement la synthèse des différentes normes proposées par l'ISO pour les couches réseau et transport. Il s'agit de (Nussbaumer 87).

Pour décrire les protocoles IP, ICMP et TCP nous avons essentiellement consulté les spécifications (RFC 791), (RFC 792) et (RFC 793) qui ont été définies par le DOD pour ces protocoles.

Signalons encore qu'un mémoire (Poncelet 87) a été réalisé en 1987 à l'Université Catholique de Louvain par Monsieur A. Poncelet. Ce mémoire est consacré principalement à la description des protocoles d'application du modèle DoD et de leur implémentation dans la version 4.2BSD. Ce mémoire et notre travail sont largement complémentaires dans la mesure où nous décrivons des mécanismes utilisés dans les protocoles IP, ICMP et TCP pour fournir un service de communication tandis que le mémoire de Monsieur A. Poncelet présente la manière dont les protocoles d'application utilisent ce service de communication pour satisfaire les besoins des utilisateurs. En fait, ensemble, ces deux mémoires fournissent une vue assez complète de l'implémentation de la version 4.2BSD du modèle DoD.

3. Qualités attendues de ce travail

Tout au long de la réalisation de ce mémoire, nous allons poursuivre les trois objectifs suivants.

- Nous allons d'abord chercher à décrire de manière très précise les fonctionnalités associées à chacun des trois protocoles qui forment le coeur du modèle DoD ainsi que les mécanismes utilisés pour y parvenir. En particulier, pour le problème de l'acheminement des messages entre les différents ordinateurs hôtes, nous devrons cerner la part que prend chaque protocole dans les aspects de routage et d'adressage de ces messages.

- Nous allons également fournir la description la plus détaillée possible des interactions entre les différents protocoles qui constituent le modèle DoD.

- Nous allons enfin essayer, chaque fois que cela est possible, de compléter les solutions proposées dans le modèle DoD par celles existant dans d'autres architectures et, en particulier, celles choisies dans le modèle OSI.

En résumé, comme ce mémoire est avant tout un travail descriptif, sa présentation devra donc être soignée et son exposé précis.

Notre apport dans ce travail se situe à différents niveaux. Lors de la description de la normalisation DoD, nous avons essentiellement cherché à décrire la démarche suivie par le DoD et à la comparer à celle adoptée par l'ISO pour son Modèle de Référence OSI. Lors de la description du protocole IP, nous avons essayé de décortiquer les mécanismes de routage et d'adressage associés à ce protocole. Ceci implique également une étude très détaillée du rôle des passerelles dans le modèle TCP/IP. Enfin, notre apport le plus important se situe, sans conteste, au niveau de la description des interactions entre les différents protocoles non seulement à l'intérieur d'un ordinateur hôte ou d'une passerelle, mais aussi avec des protocoles définis dans d'autres architectures.

4. Explication du plan du mémoire

Nous allons organiser l'étude de la normalisation DoD de la manière qui suit.

Au chapitre 1, nous présenterons le modèle DoD. En particulier, nous décrirons le processus qui a abouti à la définition de la structure du modèle DoD. Nous comparerons ensuite ce processus à celui suivi par l'ISO pour définir le Modèle de Référence OSI.

Dans les trois chapitres qui suivent, nous étudierons les trois protocoles qui sont spécifiques au modèle DoD et qui permettent aux ordinateurs hôtes de s'échanger des données. Il s'agit des protocoles IP, ICMP et TCP. Un protocole est défini comme un ensemble de règles et de formats qui déterminent la communication entre modules associés à ce protocole. Le concept de module est utilisé dans le modèle DoD pour désigner l'élément actif qui, à l'intérieur d'un ordinateur hôte ou d'une passerelle, implémente le protocole.

En fait, ce que nous allons réaliser dans ces trois chapitres, c'est l'étude des relations horizontales entre modules répartis entre différents ordinateurs mais associés chaque fois à un protocole particulier. Ce que nous entreprendrons dans les deux chapitres suivants, c'est la description des relations verticales entre les modules présents dans un même système mais associés à des protocoles différents. Ces relations sont décrites au moyen d'interfaces. Au chapitre 5, nous décrirons les interfaces conceptuelles du modèle DoD, c'est-à-dire les interfaces qui sont indépendantes d'une implémentation particulière. Au chapitre 6, nous décrirons les interfaces du modèle DoD définies dans la version 4.2BSD.

En annexe A, nous présenterons le Modèle de Référence dans ses aspects théoriques. Nous y retrouverons donc une liste de définitions de concepts qui servent de point de départ à la spécification des normes proposées par l'ISO. En annexe B, nous allons reprendre la liste des principes utilisés par l'ISO pour définir les sept couches du modèle OSI. En annexe C, nous donnerons l'explication fournie par l'ISO qui résume comment ces sept couches ont été choisies.

Enfin, nous avons repris dans les annexes restantes les formats des en-têtes des messages échangés par les principaux protocoles étudiés. Ainsi,

- en annexe D, nous trouverons le format de l'en-tête du datagramme IP;
- en annexe E, ce sont les formats des unités de données échangées dans le protocole ISO 8473 qui seront décrits;
- enfin, nous terminerons en annexe F avec le format de l'en-tête du segment TCP.

CHAPITRE 1

LA NORMALISATION DOD COMPAREE A LA NORMALISATION ISO

1.1. Introduction

L'objet de ce chapitre est de présenter l'architecture normalisée par le DoD dans le domaine de l'interconnexion de réseaux hétérogènes et de comparer cette architecture au modèle OSI défini par l'ISO.

L'architecture adoptée par le DoD repose sur une structure hiérarchique composée de quatre niveaux. La figure 1.1. représente cette structure.

Le niveau 0 est celui de l'**accès au réseau**. Il est spécifique au réseau. Il gère le transport des paquets sur différents réseaux parmi lesquels les réseaux Ethernet et Arpanet sont les plus connus. Ce niveau ne fait pas, à proprement parler, partie du cadre de la normalisation DoD (Pujolle 87, tome 2, p. 254).

La partie très spécifique au modèle DoD concerne les niveaux 1 et 2. Le niveau 1 est le niveau **inter-réseaux**, aussi appelé niveau internet. Il fournit un service d'échange de blocs de données entre un ordinateur hôte source et un ordinateur hôte destination situés éventuellement sur des réseaux éventuellement différents mais interconnectés(RFC 791, p. 1). Ce niveau regroupe les protocoles IP et ICMP étudiés respectivement aux chapitres 2 et 3.

Le niveau 2 est le niveau **bout en bout**. Il est orienté application, c'est-à-dire qu'il gère l'échange de messages entre programmes qui s'exécutent sur des ordinateurs hôtes différents. Ce niveau comprend le protocole TCP qui fournit, pour l'échange de ces messages, un service de type circuit virtuel ainsi que le protocole UDP (User Datagram Protocol) qui offre un service de type datagramme (RFC 793, p. 1 et RFC 768, p. 1).

Le niveau 3 est le niveau **applications**. Il réunit les protocoles d'application qui utilisent les services de communication fournis par les protocoles TCP et UDP. Parmi ceux-ci, nous trouvons le protocole SMTP (Simple Mail Transfer Protocol) qui fournit un service de messagerie électronique, le protocole FTP (File Transfer Protocol) qui permet le transfert de fichiers et TELNET, un protocole de terminal virtuel. Ces trois protocoles utilisent le service circuit virtuel de TCP, tandis que le protocole TFTP (Trivial File Transfer Protocol) et le protocole de serveur de noms utilisent le service datagramme de UDP.

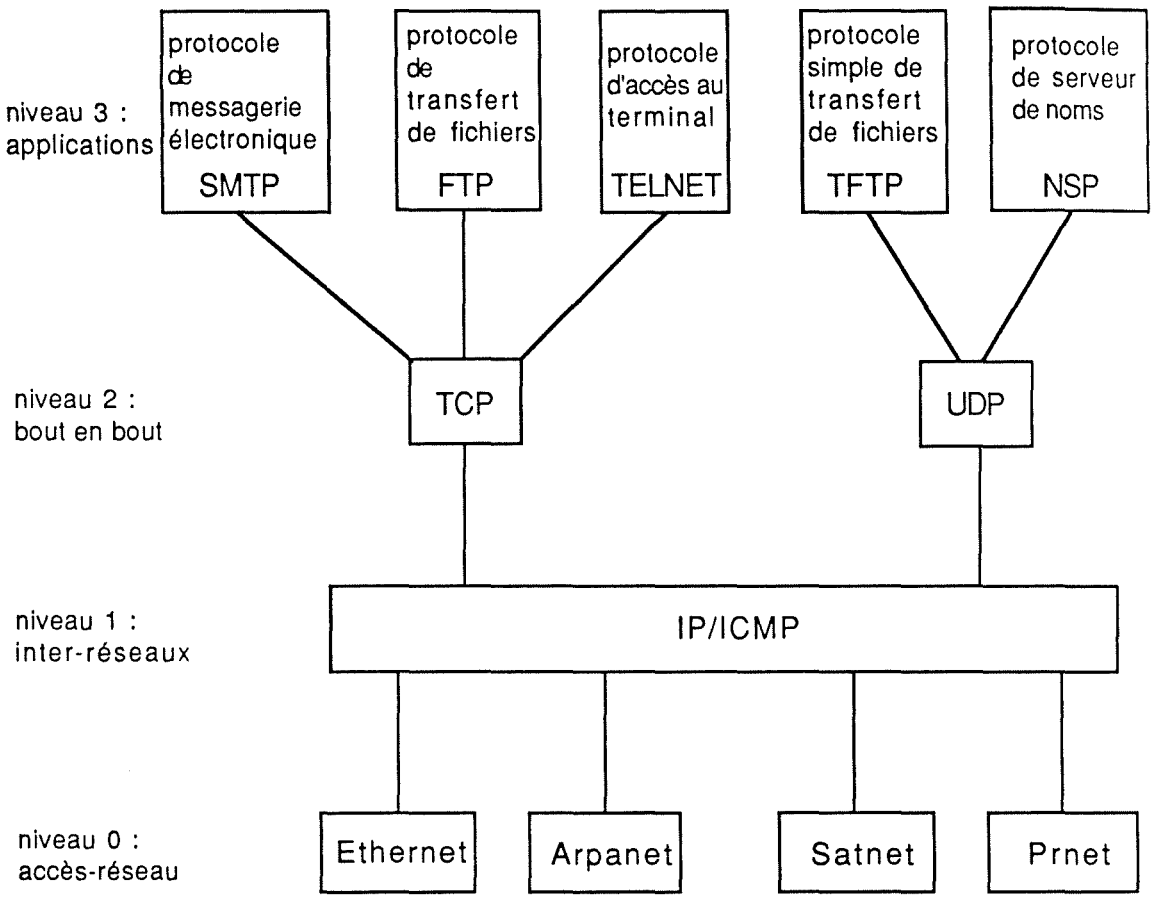


Fig. 1.1. - Architecture DoD pour l'interconnexion de réseaux hétérogènes.

Dans le reste de ce chapitre, nous allons d'abord énumérer les principales contraintes imposées par le Département de la Défense Américain pour l'élaboration de l'architecture DoD. Ceci sera l'objet de la section 1.2. A la section 1.3., nous continuerons la description du modèle DoD que nous venons ici d'entreprendre de manière à mettre en évidence les choix de conception qui ont été effectués pour honorer ces contraintes. A la section 1.4., nous présenterons la démarche suivie par l'ISO pour définir les architectures DoD et ISO.

1.2. Les contraintes du DoD

Les contraintes imposées par le Département de la Défense sur les réseaux de communication de données sont de deux natures (Selvaggi 83, p. 320).

L'ensemble des premières contraintes est bien sûr de nature militaire. Ces contraintes ne sont toutefois pas spécifiques à un département de défense. Il s'agit des contraintes suivantes (Selvaggi 83, p. 320):

1. une bonne résistance aux différentes pannes (*survivability and robustness*),
2. une grande disponibilité des moyens de communication (*avaibility*),
3. des mécanismes de sécurité (*security*),
4. une résistance au sabotage (*resistance to sabotage and electronic countermeasure*),
5. un interfaçage efficace à de nombreux réseaux différents,
6. un interfaçage avec les organisations tactiques (*interface with the tactical community*),
7. des mécanismes de priorité et de préemption (*precedence and preemption*),
8. une grande vitesse et précision (*speed and accuracy*),
9. la capacité d'opérer en mode diffusion (*capable of operating in the broodcast mode*),
10. la possibilité d'extension et de mise-à-jour aisée,
11. la capacité d'accepter des variations importantes de trafic.

La seconde nature des contraintes (Selvaggi 83, p. 321) vient de la politique adoptée par le Département de la Défense d'utiliser les standards commerciaux existant dans la plus large mesure possible, exception faite toutefois des standards commerciaux qui compromettraient les exigences militaires.

De toutes les contraintes qui viennent d'être énoncées, c'est l'exigence d'un interfaçage efficace à de nombreux types de réseau qui a amené les développements les plus importants dans le modèle DoD. Ce problème d'interfaçage est aussi appelé interopérabilité des réseaux hétérogènes. Le but poursuivi est de permettre l'échange de paquets de données entre des réseaux de caractéristiques internes et de technologies différentes. Cerf et Cain (Cerf 83, p. 309) de même que Selvaggi (Selvaggi 83, p. 324) donnent des listes non exhaustives de réseaux que le modèle DoD doit pouvoir au moins supporter. En voici les principaux :

- tous les réseaux militaires américains tels que ARPANET, WIN (World Wide Military Comman and Control System International Net), DDN (Defense Data Net), COINS (COmmunity Intelligence Network), ...
- les réseaux radio PRNET (Packet Radio NET), SRI, SAC ...
- les réseaux satellite SATNET (SATellite NET), MATNET (Navy Mobile Access Terminal NET, WB NET (Wide Band NET) , ...
- les réseaux locaux ETHERNET, CHAOSNET, MITREBUS, ...
- les réseaux X.25, TELENET, ...

1.3. Les choix conceptuels de DoD

Pour réaliser ces objectifs, c'est l'architecture en quatre niveaux décrite en introduction qui a été choisie.

Les réseaux que le DoD se propose d'interconnecter présentent des différences importantes au niveau du type de service offert (datagramme ou circuit virtuel), des formats des paquets, des formats des adresses, des performances, des technologies, etc. Pour interconnecter cette large variété de réseaux, le modèle DoD va imposer peu de contraintes à chaque réseau particulier. Seulement trois contraintes ont été retenues pour le niveau accès-réseau. Ces contraintes sont les suivantes :

- chaque réseau particulier doit au moins fournir un service datagramme avec un taux de perte inférieur à 5 % (RFC 1009, p. 59);
- chaque réseau particulier doit pouvoir transmettre des paquets de données contenant au moins 1000 bits dans leur champ de données (Cerf 78, p. 27);
- chaque réseau particulier doit permettre à tout ordinateur hôte d'envoyer des paquets à plusieurs ordinateurs hôtes du réseau dans un délai de quelques dizaines de millisecondes (Cerf 78, p. 28).

La principale mission assignée au niveau 1 est d'obtenir l'interconnexion des différents réseaux. Pour y arriver, le modèle DoD a choisi d'utiliser un mécanisme unique de transport de données. Ce mécanisme, c'est le protocole IP. Ce **protocole IP** regroupe toutes les fonctions nécessaires pour transmettre une suite d'octets, appelée **datagramme**, entre un ordinateur hôte source et un ordinateur hôte destination connectés éventuellement à des réseaux distincts. Il ne dispose toutefois pas de mécanismes qui augmentent la fiabilité de la transmission des données de bout en bout, ou qui contrôlent le flux des datagrammes, ou encore qui règlent le séquencement des datagrammes. Le service offert par le protocole IP est donc non fiable (RFC 791, p. 1).

Pour obtenir l'interopérabilité des réseaux, le modèle DoD a également choisi d'utiliser des **passerelles**, responsables non seulement du choix de l'itinéraire à suivre par les datagrammes quand ceux-ci doivent traverser différents réseaux, mais aussi de la transmission de ces datagrammes sur chaque réseau particulier. Une passerelle connecte deux ou plusieurs réseaux. Pour transmettre les datagrammes sur un réseau particulier, la passerelle utilise le protocole de transmission propre à ce réseau. L'ensemble des réseaux particuliers connectés entre eux par des passerelles forme un **catenet** (Postel 81, p. 262). La figure 1.2. nous montre un exemple de catenet comprenant trois réseaux particuliers A, B et C et quatre passerelles P1, P2, P3 et P4. Les passerelles P1, P2, P3 connectent, chacune,deux réseaux tandis que la passerelle P4 relie les trois réseaux. A chaque réseau est connectée également une série d'ordinateurs hôtes.

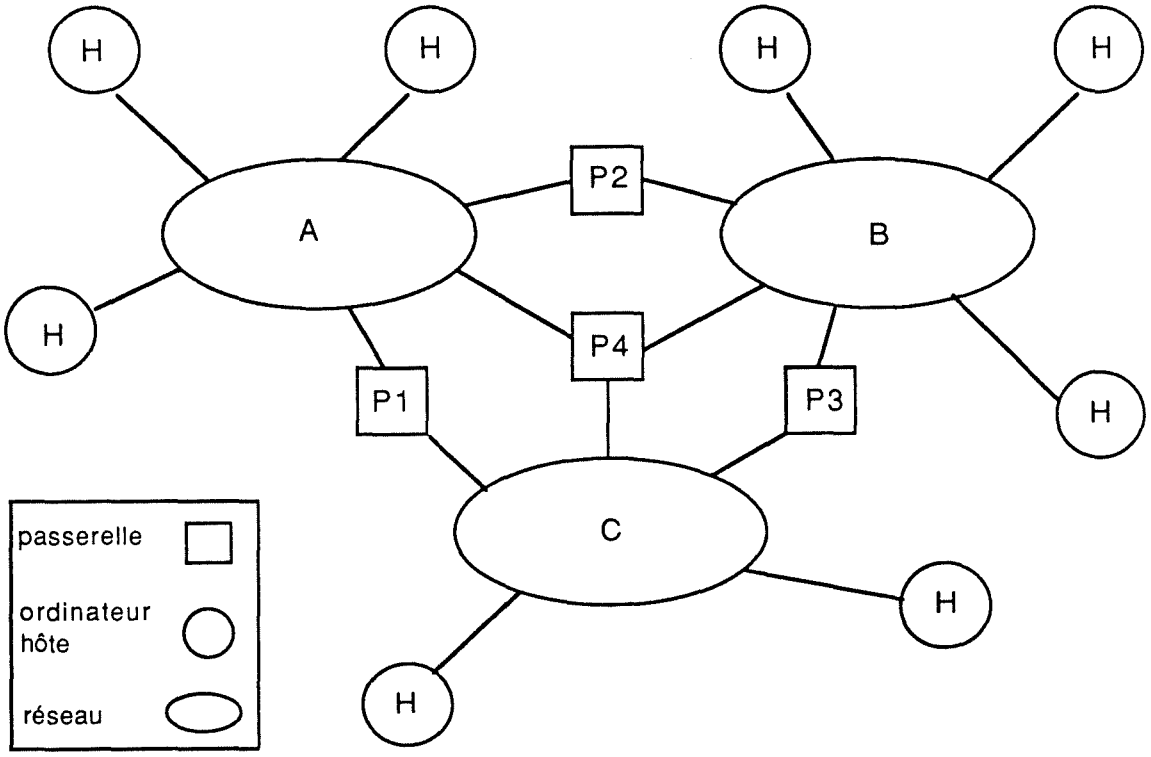


Fig. 1.2. - Exemple de catenet.

Tous les ordinateurs hôtes implémentent les protocoles du niveau 0 au niveau 3 tandis que les protocoles du niveau le plus haut qui sont implémentés sur les passerelles, sont généralement les protocoles IP et ICMP du niveau 1.

Le protocole IP implémente trois fonctions de base : l'adressage, le routage et la fragmentation. **L'adresse internet** est une adresse de longueur fixe. Elle permet d'identifier les ordinateurs hôtes et les passerelles du catenet. Cette adresse internet est utilisée pour acheminer les datagrammes à leur destination. Quand un datagramme doit traverser un réseau dont la taille maximale du champ des données des paquets est inférieure à la taille du datagramme, le protocole IP fragmentera le datagramme en morceaux plus petits. Le protocole IP traite chaque nouveau datagramme de manière indépendante des autres datagrammes. Ils pourront donc suivre des itinéraires différents avant d'arriver à destination (RFC 791, p. 2).

Le **protocole ICMP** est considéré comme partie intégrante du protocole IP bien qu'il soit souvent placé au dessus de IP (RFC 1009, p. 59). Ceci peut s'expliquer par le fait que le protocole ICMP utilise le service de transport d'IP pour communiquer ses messages. Le protocole ICMP est chargé de renseigner l'ordinateur hôte source des problèmes qui surviennent aux datagrammes lors de leur transmission. Il peut également fournir à l'ordinateur hôte source des conseils ou des renseignements concernant le fonctionnement du catenet (RFC 792, p. 1). Le protocole ICMP ne garantit toutefois pas que, si le datagramme n'a pas été livré à l'ordinateur hôte destination, l'ordinateur hôte source en sera informé. Seuls les problèmes détectés par le protocole IP et pour lesquels un message ICMP est prévu, seront communiqués à l'ordinateur hôte source.

D'autres protocoles utilisent également le protocole IP pour transporter des informations relatives à la gestion du catenet. Il s'agit des protocoles GGP (Gateway to Gateway Protocol), RIP[1] (Routing Information Protocol), EGP[2] (Exterior Gateway Protocol). Tous ces protocoles aident les passerelles dans leur fonction de routage des datagrammes.

Alors que le modèle DoD recherchait, avec le protocole IP, l'interopérabilité des réseaux, il va essayer, avec le **protocole TCP**, de satisfaire les exigences de résistance aux incidents et de disponibilité en cas de congestion. Ce protocole TCP offre un service de communication fiable entre processus appartenant à des ordinateurs reliés à des réseaux éventuellement distincts mais interconnectés. Ce service est de type circuit virtuel, c'est-à-dire orienté connexion. Le protocole TCP suppose qu'il dispose de protocoles de niveau inférieur qui lui fournissent un service datagramme semblable à celui que le protocole IP propose (RFC 792, p. 2).

Chaque ordinateur hôte qui veut utiliser un réseau interconnecté basé sur le modèle DoD doit disposer d'un module IP et d'un module TCP. L'unité de données échangée entre deux modules TCP s'appelle le **segment TCP**.

---

[1] Cfr. paragraphe 2.3.3.5. pour la description de ce protocole.

[2] Cfr. point 2.3.4.3.5. pour la description de ce protocole.

Pour pouvoir offrir une communication fiable entre deux ordinateurs, le protocole TCP utilise les mécanismes suivants :

- l'acquittement positif et la retransmission,
- le contrôle de flux,
- le séquencement,
- le multiplexage,
- les connexions,
- les priorités et la sécurité (RFC 792, p. 3).

Tous ces mécanismes sont généralement connus sauf peut-être la technique du multiplexage. Le **multiplexage** veut permettre à plusieurs processus d'un même ordinateur de communiquer simultanément par le même module TCP. Pour y parvenir, le protocole TCP fournit un ensemble de portes dans chaque ordinateur. Ces **portes** permettent au module TCP d'identifier les processus et d'associer les données, reçues ou à envoyer, à un processus. Ce numéro de porte concaténé à l'adresse internet forme un **socket**. Une paire de sockets identifie une **connexion** (RFC 792, p. 5, p. 82).

Dans le modèle DoD, on a aussi voulu que certains protocoles d'application puissent encore disposer d'un service de type datagramme au niveau 2. Ceci explique la présence du **protocole UDP** (RFC 768) qui, par rapport au protocole IP, n'ajoute que la fonction de multiplexage.

Les protocoles TCP et UDP ne présentent aucun intérêt s'ils ne sont pas utilisés par des logiciels d'application. Le modèle DoD a choisi d'associer à chaque application un protocole particulier de niveau 3 qui, à son tour, utilisera un service de transport du niveau 2 fourni, soit par TCP, soit par UDP. Ainsi, pour l'application de transfert de fichiers, on retrouve, au niveau 3, les **protocoles FTP** et **TFTP**. Le premier utilise le protocole TCP tandis que le second utilise le protocole UDP. Le niveau 3 comprend également un protocole de terminal virtuel, **TELNET** qui utilise TCP, un protocole de messagerie, **SMTP** (Simple Mail Transfer Protocol) qui utilise TCP, un protocole de serveur de nom, **NSP** (Name Server Protocol) qui utilise UDP et bien d'autres protocoles. Signalons que le protocole NSP renvoie l'adresse internet d'un ordinateur hôte à partir de son nom.

Dans cette section, nous venons de dégager les quelques idées maîtresses qui ont régi la conception du modèle DoD. Ce modèle, composé de quatre niveaux, s'articulent autour des protocoles IP et TCP. Le protocole IP vise essentiellement l'interopérabilité des réseaux tandis que le protocole TCP cherche à offrir un service de communication robuste aux incidents. Dans la section suivante, nous allons présenter la démarche suivie par l'ISO.

1.4. Le modèle OSI

Tout comme le modèle DoD, le modèle OSI essaie d'apporter une solution aux problèmes que pose l'interconnexion de réseaux. Ces problèmes peuvent provenir des différences entre les réseaux connectés au niveau de leur protocole, leur service, leur interface, leur format de paquets, etc. Ils peuvent aussi surgir des ordinateurs reliés, suivant leur manière d'organiser une session terminale, de protéger ou de référencer leurs fichiers, de soumettre des travaux, etc. Pour remédier à ces problèmes, les modèles DoD et OSI proposent des architectures hiérarchiques qui ne résultent pas des efforts d'un constructeur particulier.

John D. Day et Hubert Zimmermann expliquent dans l'article de référence qu'ils ont consacré au modèle OSI (Zimmermann 83) la démarche suivie par l'ISO lors de l'élaboration de ce modèle. Ils y déclarent ceci. "*Dans OSI, le problème a été abordé de manière top-down, en partant d'une description d'un haut niveau d'abstraction qui impose peu de contraintes, et procédant à des descriptions de plus en plus raffinées, elles intégrent des contraintes de plus en plus strictes. Dans le monde OSI, on reconnaît trois niveaux d'abstraction : l'architecture OSI, les spécifications de service et les spécifications de protocole.*

L'architecture OSI est le plus haut niveau d'abstraction du modèle OSI" (Zimmermann 83, p. 1334). Cette architecture est spécifiée très scrupuleusement par la norme ISO 7498 et ses annexes. Cette norme décrit les fondements du **Modèle de Référence OSI**. Elles se composent de deux parties. La première décrit les éléments de cette architecture tandis que la seconde énumère les services et les fonctions des sept couches choisies par l'ISO. Nous n'allons pas définir ici les éléments de cette architecture. Une description des principaux d'entre eux est donnée en annexe A.

L'idée de base de la division en couches est que chaque couche ajoute de la valeur aux services fournis par l'ensemble des couches inférieures de telle sorte que la couche la plus haute dispose de l'ensemble complet des services nécessaires pour exécuter les applications. La division en couches morcelle simplement le problème général en morceaux plus petits (Zimmermann 83, p. 1336).

Un autre principe de base associé à la division en couches est d'assurer l'indépendance de chaque niveau en définissant les services fournis par une couche à la couche immédiatement supérieure indépendamment de la manière dont ces services sont réalisés. Une couche peut modifier son fonctionnement interne sans avoir d'effets sur la couche immédiatement supérieure si elle continue à offrir le même service (Zimmermann 83, p. 1336).

Dans la mesure où il est théorique, le modèle décrit en annexe A permet d'envisager un nombre illimité de possibilités et notamment un nombre arbitraire de couches. L'ISO justifie son architecture en 7 couches en énonçant d'une part, les 13 principes (ISO 7498, pp. 39-40) qui ont été utilisés pour déterminer ces sept couches et en présentant d'autre part, une explication intuitive de la manière dont ces sept couches ont été choisies (ISO 7498, pp. 74-75). Nous avons repris en intégralité les 13 principes en annexe B et l'explication intuitive en annexe C.

L'architecture OSI décrite par la norme ISO 7498 constitue, rappelons-le, le plus haut niveau d'abstraction du Modèle de Référence OSI. Le niveau intermédiaire d'abstraction est occupé par les spécifications des services OSI. Ce niveau permet de mettre en évidence la distinction qui existe entre utilisateurs et fournisseurs de service (ISO/DIS 8509, pp. 3-4). Leur définition est reprise en annexe A parce qu'elle suit logiquement la description des éléments de l'architecture OSI. Citons, à titre d'exemple, quelques normes qui spécifient les services ISO. Le projet ISO/DIS 8348 décrit le service réseau en mode connexion tandis que son addendum 1 le définit pour une transmission en mode sans connexion. De la même manière, le projet ISO/DIS 8072 décrit le service transport en mode connexion alors que son addendum 1 définit un service transport en mode sans connexion. La norme ISO 8326 définit le service de base session en mode connexion. L'avant-projet ISO/DP 8831 définit le service de transfert et de manipulation de travaux à distance, etc.

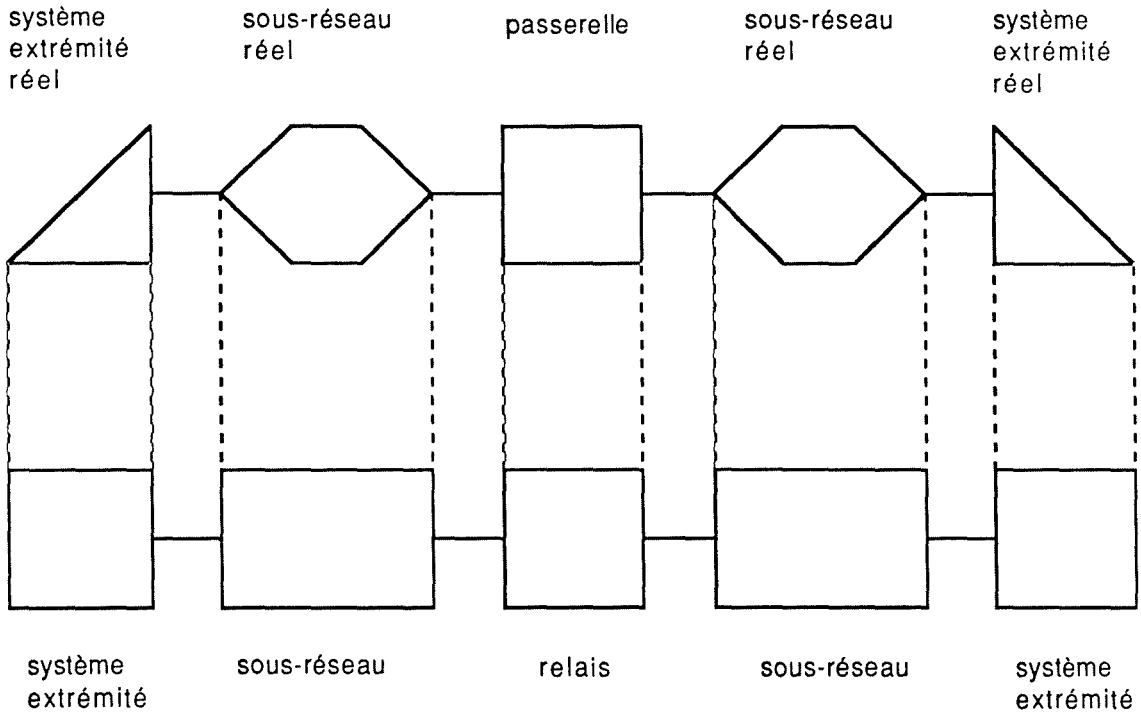


Fig. 1.3. - Exemple de connexion de deux systèmes réels à travers deux sous-réseaux réels.

Le niveau d'abstraction le plus bas est celui des spécifications des protocoles OSI. Chaque spécification de protocole définit très précisément quelle information de contrôle doit être envoyée et quelles procédures doivent être utilisées pour interpréter cette information de contrôle (Zimmermann 83, p. 1335). Les normes portant sur les protocoles sont au moins aussi nombreuses que les normes définissant les services. Citons, pour la couche réseau, le projet ISO/DIS 8473 décrivant le protocole de transmission de données fournissant le service de réseau en mode sans connexion, l'avis X.25 du CCITT qui décrit l'interface entre un

équipement terminal de traitement de données (ETTD) et un équipement de teminaison de circuits de données (ETCD) pour terminaux fonctionnant en mode-paquet, raccordés à un réseau public de transmission de données. L'ISO a rencontré depuis le début, des problèmes d'harmonisation des services de réseau, spécialement dans le cas où les systèmes d'extrémité sont connectés par l'intermédiaire d'un ou plusieurs sous-réseaux réels. "*Ce type de situation est illustré sur la figure 1.3.* (source : ISO/DIS 8648, p. 13) *où les deux systèmes d'extrémité sont connectés par l'intermédiaire de deux sous-réseaux réels, dont l'un peut être par exemple un réseau public à commutation de paquets et l'autre peut être un réseau local. Les deux sous-réseaux sont ici interconnectés par un système intermédiaire qui sert de* ***relais*** *et qui peut être par exemple une* ***passerelle.*** *La partie supérieure de la figure 1.3. représente les composants réels du système complet, et la partie inférieure indique les objets abstraits qui leur correspondent*" (Nussbaumer 87, p. 175).

"*Avec un système tel que celui de la figure 1.3., il peut se poser des problèmes délicats de compatibilité entre les protocoles de réseau mis en oeuvre par les différents sous-ensembles. Pour faciliter la solution de ce genre de problèmes, l'ISO a proposé (ISO/DIS 8648) un modèle de l'organisation interne de la couche réseau avec les objectifs suivants :*

- *simplifier l'utilisation des protocoles de réseau pour fournir un service de réseau dans des situations différentes;*
- *limiter la prolifération de protocoles de réseau avec des fonctions redoutantes;*
- *clarifier les besoins et fournir un guide pour le développement de normes de protocoles de réseau*" (Nussbaumer 87, p. 175).

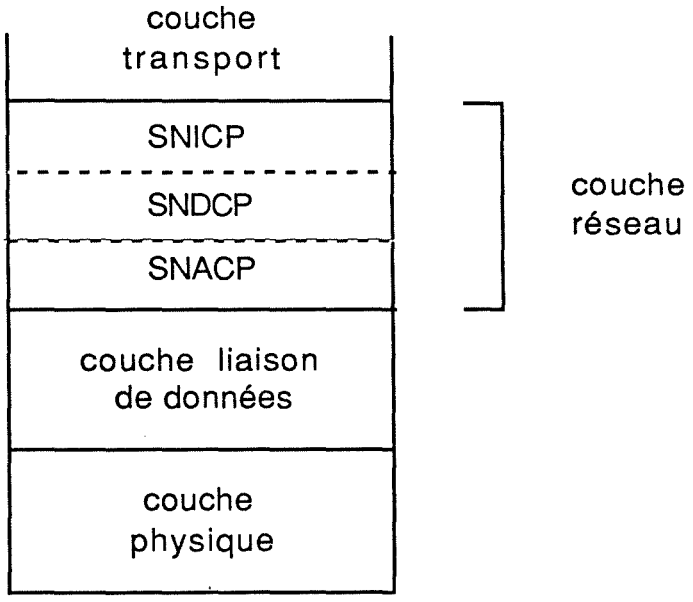


Fig. 1.4. - Organisation interne de la couche réseau.

"La structure proposée par l'ISO pour la couche réseau se compose de trois sous-couches. Elle est représentée à la figure 1.4. Au niveau le plus bas, le protocole SNACP (Subnetwork Access Protocol) assure les fonctions de réseau qui sont spécifiques à un sous-réseau réel particulier, et qui concernent en particulier le routage dans le sous-réseau. Comme le service fourni par cette sous-couche n'est en général pas celui qui est désiré, il faut le plus souvent adapter le protocole SNACP grâce à deux sous-couches supplémentaires. Au niveau le plus élevé, cette fonction d'harmonisation est prise en charge par un protocole SNICP (Subnetwork Independant Convergence Protocol) qui fournit à la couche de transport un service de réseau OSI, et qui exploite un jeu bien défini de fonctions offertes par un protocole intermédiaire appelé SNDCP (Subnetwork Dependent Convergence Protocol). On voit donc que le protocole SNICP est relativement indépendant des caractéristiques de sous-réseau réel, et que le protocole SNDCP sert essentiellement à assurer un découplage entre les caractéristiques qui sont liées au sous-réseau et l'organisation du protocole qui fournit le service de réseau à la couche de transport " (Nussbaumer 87, p. 176).

"Lorsque l'interconnexion entre deux sous-réseaux réels est réalisée au niveau de la couche réseau, la passerelle peut être organisée comme indiqué à la figure 1.5., où les sous-couches SNDCP et SNICP qui sont disposées de chaque côté harmonisent les protocoles propres aux deux sous-réseaux au niveau d'un service commun ISO. La fonction de relais qui est située à la partie supérieure de la passerelle permet d'expédier les informations reçues d'un correspondant situé dans un des réseaux vers un correspondant situé dans l'autre réseau. La passerelle assure également le routage entre les réseaux" (Nussbaumer 87, p. 176).

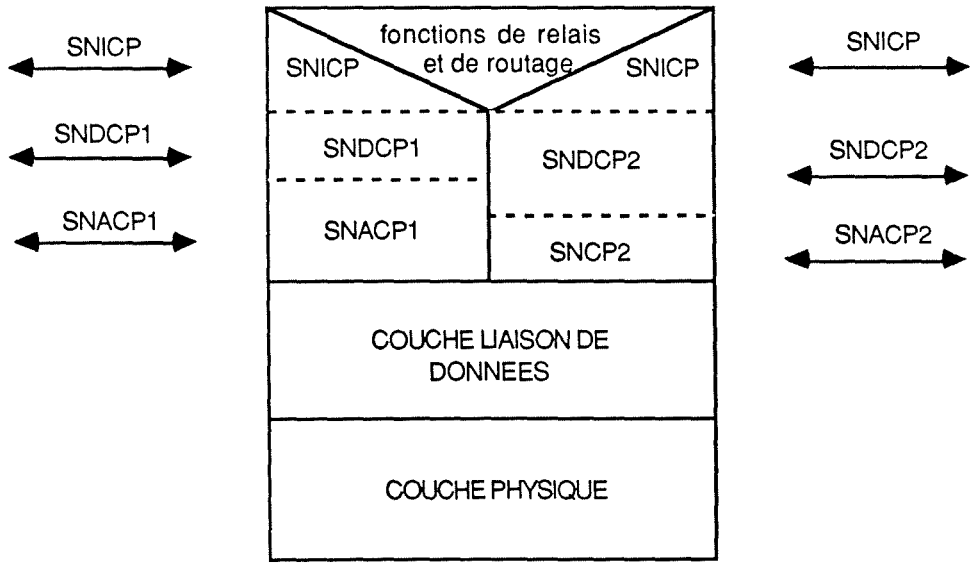


Fig. 1.5. - Relais au niveau réseau.

1.5. Comparaison des architectures DoD et OSI

Après avoir décrit sommairement les démarches suivies par l'ISO et le DoD pour définir leurs architectures, nous sommes en mesure de les comparer. Cette comparaison va s'articuler autour des quatre niveaux du modèle DoD. Avant de passer en revue chacun de ces niveaux, nous rappelons au tableau 1.1. la correspondance généralement admise entre les niveaux des deux modèles (source : Stallings 85, p. 400).

<table>
<tr><th>MODELE OSI</th><th>MODELE DOD</th></tr>
<tr><td>application</td><td rowspan="3">applications</td></tr>
<tr><td>présentation</td></tr>
<tr><td>session</td></tr>
<tr><td>transport</td><td>bout en bout</td></tr>
<tr><td>SNICP
SNDCP</td><td>inter-réseaux</td></tr>
<tr><td>SNACP</td><td rowspan="3">accès-réseau</td></tr>
<tr><td>liaison</td></tr>
<tr><td>physique</td></tr>
</table>

Tab. 1.1. - Comparaison des architectures OSI et DoD.

1.5.1. Au niveau accès-réseau

"*Le but poursuivi par OSI est de permettre à n'importe quel ordinateur situé n'importe où dans le monde de communiquer avec un autre ordinateur tant que ceux-ci respectent les normes OSI*" (Zimmermann 83, p. 1335). Ceci entraîne de la part de l'ISO, du CCITT et de l'ECMA une grande activité dans la définition de normes pour les couches 1, 2 et 3. A l'opposé, le modèle DoD essaie d'utiliser tout ce qui est disponible sur le marché comme réseaux à technique de paquet tant que ceux-ci ne remettent pas en cause les objectifs définis par le DoD. Ceci explique l'intérêt relatif manifesté par le DoD au niveau accès-réseau. A ce niveau, on peut qualifier la philosophie OSI de restrictive dans la mesure où les systèmes doivent respecter les normes OSI tandis que la philosophie DoD est résolument libérale puisque le modèle DoD vise à l'interconnexion de tout système commecialisé.

1.5.2. Au niveau inter-réseaux

Les principaux services assurés par le niveau inter-réseaux du modèle DoD et par la couche réseau du modèle OSI sont les suivants :

- indépendance par rapport aux supports de transmission sous-jacents,
- transfert de bout en bout,
- transfert transparent des informations,
- choix de la qualité de service,
- adressage de l'utilisateur du service offert.

Les services offerts et donc aussi la manière de les réaliser varient profondément d'un modèle à l'autre.

Le niveau inter-réseaux offre avec le protocole IP un service datagramme unique tandis que la couche réseau OSI fournit un service datagramme (sans connexion) analogue à celui d'IP et un service circuit virtuel (avec connexion). La préférence du DoD pour un service datagramme s'explique parce que cela permet :

- d'utiliser des passerelles plus simples qui n'ont pas à gérer des informations d'état concernant chaque circuit créé;

- de continuer les communications de bout en bout par des chemins alternatifs dans le cas où une passerelle ou un réseau venait à tomber en panne;

- de laisser une plus grande marge de manoeuvre pour définir de nouveaux protocoles autres que TCP et UDP au niveau bout en bout.

En ce qui concerne la fonction d'interopérabilité des réseaux, le modèle DoD propose une solution unique : l'interconnexion des réseaux au moyen de passerelles implémentant le protocole IP. Le modèle OSI propose, quant à lui, de nombreuses solutions, mais celles-ci ne sont pas encore toutes complètement spécifiées. Ainsi, par exemple, le projet ISO/DIS 8473 propose une solution fort analogue à celle du protocole IP, c'est pourquoi il est également appelé protocole CL-IP (Connection Less - Internet Protocol) ou ISO-IP (International Standards Organisation - Internet Protocol). Ce protocole est typiquement un protocole SNICP. Une comparaison détaillée de ces deux protocoles est réalisée au chapitre suivant. A titre d'information, nous signalons que l'article (Bauerfeld 87) présente un aperçu des tendances actuelles de l'ISO en ce qui concerne le problème de l'interconnexion des réseaux LAN (Local Area Network) et WAN (Wide Area Network).

Enfin, il reste à remarquer que le protocole IP du DoD recouvre les fonctions des protocoles SNDCP et SNICP dans la mesure où il doit également harmoniser les services offerts par les différents réseaux.

1.5.3. Au niveau bout en bout

Le niveau bout en bout du DoD ainsi que la couche transport de l'ISO offrent deux types de service transport : un service transport orienté connexion et un autre orienté sans connexion. Le service transport sans connexion est prévu pour les applications temps réel portant sur l'acquisition et la distribution de données. Le service avec connexion est bien adapté à la plupart des autres applications informatiques; il est de loin le plus utilisé.

Pour assurer ces deux services, le modèle DoD a spécifié deux protocoles : le protocole TCP (RFC 793) pour offrir le service circuit virtuel et le protocole UDP (RFC 768) pour le service datagramme. Le modèle OSI, pour sa part, en a défini cinq (ISO/DIS 8073) pour offrir un service de transport orienté connexion et un (ISO/DIS 8073/ADD1) pour le service sans connexion. Cette simplicité du modèle DoD repose sur le fait que le niveau inter-réseaux offre un service unique de transport non fiable. Par contre, dans le modèle OSI, la couche transport est confrontée à une diversité de services offerts par la couche réseau.

Pour son service transport orienté connexion, l'ISO a été ainsi amené à définir trois types de service de réseau qui sont repérés par les lettres A, B et C et dont les caractéristiques sont reprises au tableau 1.2. Cette classification permet de déterminer la classe de connexion transport qui doit être utilisée pour assurer un service satisfaisant. Dans ce contexte, une **erreur résiduelle** correspond à une unité de données de protocole qui est perdue, dupliquée ou corrompue par des erreurs et ceci sans que la couche réseau en soit consciente. Les **erreurs signalées** correspondent à des incidents qui se sont produits dans le réseau et qui ont provoqué des erreurs irrécupérables au niveau du réseau mais qui sont signalés à la couche transport.

types de réseaux	taux d'erreurs résiduels	taux d'incidents signalées
A	acceptable	acceptable
B	acceptable	inacceptable
C	inacceptable	inacceptable

Tab. 1.2. - Types de réseaux.

Le type A correspond à un réseau fiable; le type B correspond à un réseau qui ne présente que des incidents signalés; le type C définit un réseau non fiable susceptible de présenter des défauts non signalés.

Les entités de transport doivent connaître le type de réseau dont elles disposent pour mettre en oeuvre les procédures qui leur permettent d'assurer la qualité de service requise par l'utilisateur. Pour simplifier le choix de ces procédures, celles-ci ont été regroupées en cinq classes, numérotées de 0 à 4. Ces classes sont les suivantes :

- la classe 0 est la classe la plus simple. Elle autorise les liaisons de transport sans amélioration propre de qualité de service. Cette classe s'adapte mieux aux réseaux de type A;

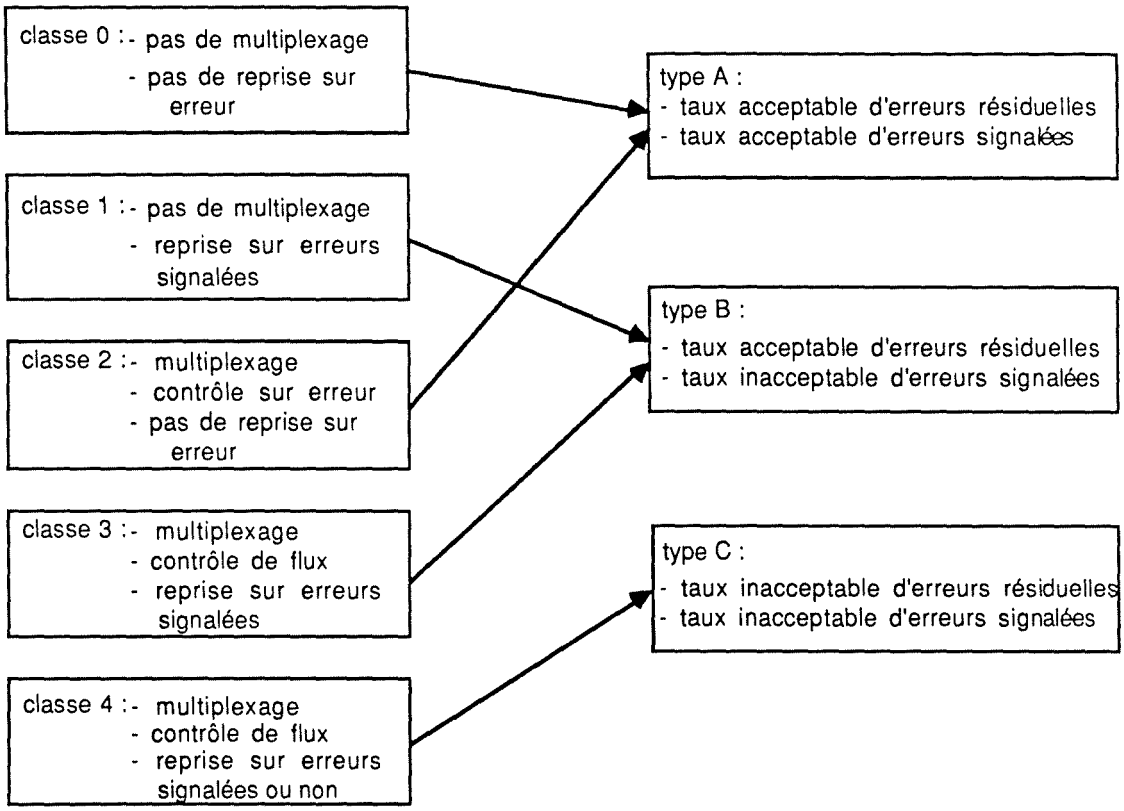
- la classe 1 comprend les fonctionnalités de la classe 0. Elle fournit en outre la reprise sur erreurs signalés par la couche réseau. En ce sens, elle s'adapte bien aux réseaux de type B;

- la classe 2 offre une possibilité de multiplexage de plusieurs connexions de transport sur une connexion de réseau. Lors de l'établissement d'une connexion de cette classe, il est possible de recourir ou non à un contrôle de flux explicite pour éviter les problèmes de congestion sur la connexion réseau. Cette classe sans contrôle d'erreurs est adaptée aux réseaux de type A;

- la classe 3 conjugue les propriétés de la classe 2 pour ce qui concerne le multiplexage et de la classe 1 pour ce qui concerne les reprises sur erreurs. Elle s'adapte plus particulièrement aux réseaux de type B;

- la classe 4 reprend les propriétés de la classe 3 avec en plus, la reprise sur erreurs non signalées par le réseau. Elle s'adapte donc parfaitement au réseau de type C.

Le tableau 1.3. résume les principales caractéristiques de chaque classe de protocole ainsi que les types de réseau auxquels elle s'adresse. Il est cependant nécessaire de signaler que ces affectations transport-réseau sont des affectations "naturelles" mais que rien n'empêche d'utiliser un protocole de transport de classe 4 avec un réseau de type A. Ceci peut avoir lieu chaque fois qu'une qualité de service élevée au niveau du taux d'erreur sera nécessaire à l'utilisateur du service de transport.



Tab. 1.3. - Classes de transports et types de réseaux.

1.5.4. Au niveau applications

A ce niveau, plus aucune comparaison n'est encore possible. Face à une application particulière, le modèle DoD propose une solution unifiée alors que le modèle OSI distingue encore les aspects relatifs à l'organisation de la session, à la présentation des données et à la spécificité de l'application.

1.5.5. Evolution

Devant la multitude des normes définies dans le modèle OSI pour les couches 1, 2, 3 et 4, les utilisateurs ont décidé de faire entendre leur voix par l'intermédiaire de deux groupes : le MAP (Manufacturing Automation Protocol) et le TOP (Technical and Office Protocol). Le projet MAP a été lancé en 1980 par General Motors dans le but de définir un standard de réseau local pour les milieux industriels. Le projet TOP, quant à lui, a été développé par Boeing et est destiné aux environnements bureautiques et techniques. Ces deux projets ont choisis parmi les normes internationales, celles qui permettent de satisfaire au mieux leur besoin. Cela les a amené à adopter une structure comparable à celle du modèle DoD, à savoir le protocole CL-IP à la couche réseau et le protocole ISO classe 4 à la couche transport.

Ces deux projets nous proviennent des Etats-Unis. En Europe, on a également entrepris de définir des standards fonctionnels. Ces standards sont des services basés sur les normes ISO/OSI et sur les recommandations du CCITT. Ils sont définis par le CEN/CENELEC et par le CEPT. Par ailleurs, la Communauté Européenne coordonne les efforts de nombreux pays européens dans ce domaine. Ainsi, le projet baptisé COSINE (Cooperation for Open Systems Interconnection Networking in Europe) est supporté par la CEE et par l'organisation RARE (Réseaux Associés pour la Recherche Européenne). Le projet se base sur les protocoles X.25 pour les couches 1, 2 et 3 et un protocole ISO classe 0 pour la couche transport.

Comme on le voit, le débat entre les partisans d'un service réseau avec connexion et ceux d'un service réseau sans connexion continue.

CHAPITRE 2

LE PROTOCOLE IP

2.1. Introduction

Le protocole IP occupe dans l 'architecture hiérarchique du modèle DoD une place centrale. Il est en effet l'unique protocole situé au dessus des protocoles tels que Ethernet du niveau accès-réseau. De plus, il est utilisé conjointement par les protocoles TCP et UDP du niveau bout en bout, et par le protocole ICMP de niveau inter-réseaux.

Sur chaque ordinateur hôte connecté au réseau ainsi que sur chaque passerelle, se trouve au moins un module IP qui implémente le protocole IP. Les principales tâches assignées à ces modules sont les suivantes :

- la désignation du chemin à suivre par les datagrammes sur base de leurs champs d'adresse[1] ;
- la fragmentation des datagrammes trop longs et leur réassemblage à l'ordinateur hôte destination.

Pour obtenir une description intuitive de l'agencement de ces principales fonctions, nous renvoyons le lecteur au document qui spécifie le protocole IP (RFC 791, pp. 5-6).

Nous allons étudier le protocole IP de la manière suivante : à la section 2.3., nous décrirons deux mécanismes étroitement liés, ceux de l'adressage et du routage des datagrammes; à la section 2.4., nous présenterons les mécanismes de fragmentation et de réassemblage des datagrammes; à la section 2.5., nous passerons en revue les autres fonctions du protocole IP.

Pour chacune des fonctions présentées dans ce chapitre, nous étayerons les solutions du modèle DoD par celles proposées par le protocole ISO 8473 en ce qui concerne la transmission de données fournissant le service de réseau en mode sans connexion (ISO/DIS 8473). A la section 2.6, nous présenterons un résumé des points communs et des points de divergence de ces deux protocoles. Avant de commencer la description du protocole IP,nous allons d'abord introduire le protocole ISO 8473 à la section 2.2.

---

[1] Il s'agit de l'adresse internet qui est définie au paragraphe 2.3.2.1.

2.2. Introduction au protocole ISO 8473

"*La norme ISO 8473 ne définit que la sous-couche de réseau qui est indépendante du sous-réseau réel utilisé. Il s'agit donc ici d'un protocole SNICP au sens où nous l'avons présenté dans le chapitre précédent, et qui ne constitue que la partie supérieure de la couche réseau*[2]*. Ce protocole devra donc dans la plupart des cas être complété par une sous-couche d'accès SNACP particulière au sous-réseau réel utilisé, et par une sous-couche SNDCP d'adaptation* " (Nussbaumer 87, p. 177).

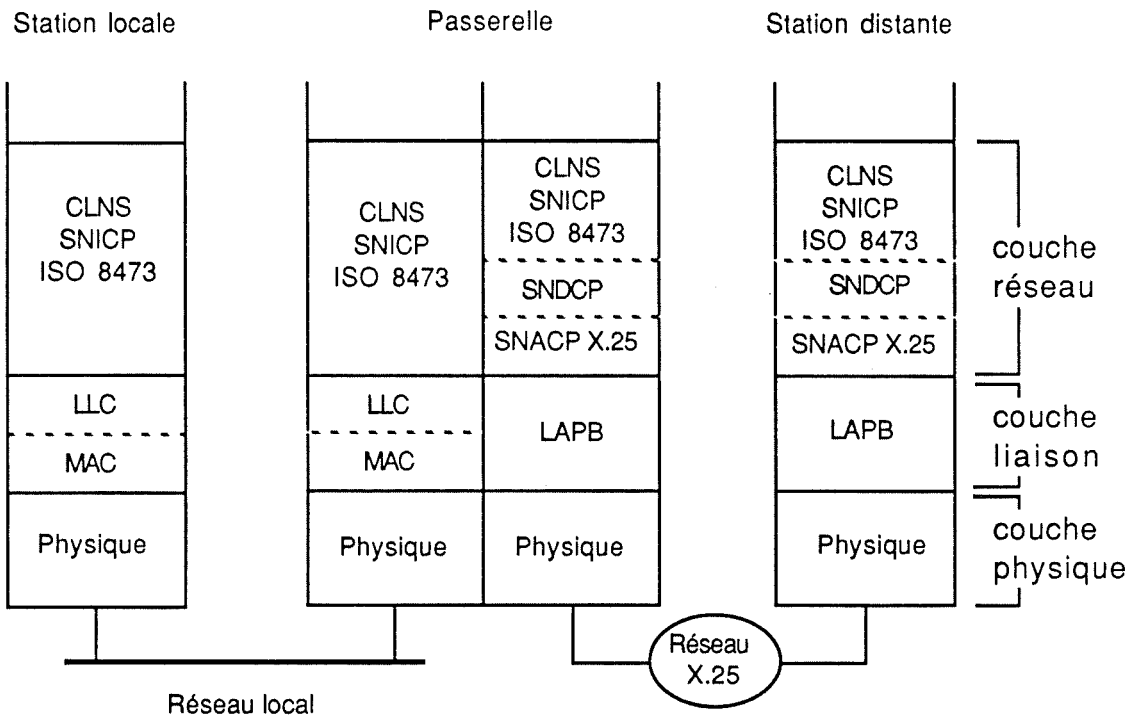


Fig. 2.1. - Exemple de mise en oeuvre de la couche de réseau sans connexion ISO 8473.

2 Cfr. Fig. 1.4. pour une représentation de cette couche.

"*L'organisation générale des protocoles avec une couche de réseau sans connexion est illustrée à la figure 2.1. dans le cas d'un réseau local exploité avec les protocoles IEEE 802. La couche liaison de données est ici divisée en une sous-couche MAC (Medium Access Control) qui est propre au type de réseau utilisé, et en une sous-couche LLC (Logical Link Control) qui regroupe les fonctions indépendantes du type de réseau. Dans sa version la plus simple, dite de type 1, le protocole de ligne LLC travaille sans connexion, avec transfert des données par expédition de trames isolées, sans numérotation de séquence. Le service ainsi assuré est pratiquement celui qui utilisé le protocole SNICP ISO 8473, à quelques détails près qui concernent les paramètres de qualité de service. Ceci permet d'assurer toutes les fonctions de la couche réseau avec le seul protocole ISO 8473*" (Nussbaumer 87, p. 178).

"*Sur l'exemple de la figure 2.1., les stations du réseau local ont la possibilité d'accéder à un réseau public X.25 par l'intermédiaire d'une passerelle. La sous-couche de réseau SNACP est donc ici le protocole X.25 de niveau paquet. Comme ce dernier ne fournit plus dans sa version 1984 de service de datagrammes, il faut alors prévoir un protocole d'adaptation SNDCP qui offre au protocole ISO 8473 un service sans connexion à partir du service avec connexion de X.25. Cette configuration est évidemment assez baroque, mais elle est justifiée ici par la nécessité d'exploiter les réseaux publics existants*" (Nussbaumer 87, p. 179).

2.3. Fonctions d'adressage et de routage des datagrammes

2.3.1. Introduction

Le modèle DoD accorde un grand intérêt aux concepts de nom, d'adresse et de route. On définit généralement le **nom** d'une ressource comme une suite de caractères qui identifie la ressource, une **adresse** comme l'endroit où se trouve la ressource et une **route** comme la manière d'atteindre cette ressource (Shoch 78, p. 280).

Au paragraphe 2.3.2., nous verrons que le protocole IP s'intéresse particulièrement à la structure des adresses ainsi qu'à leur conversion. Il laisse toutefois le problème de la conversion des noms des ordinateurs hôtes en adresses internet aux protocoles de plus haut niveau. Le protocole IP s'occupe aussi du problème de la sélection du meilleur chemin à suivre par les datagrammes. Ce problème est connu sous le nom de routage ou d'acheminement. Il sera étudié au paragraphe 2.3.3. En relation avec cette fonction de routage, les passerelles jouent un rôle déterminant. Ce rôle sera étudié au paragraphe 2.3.4.

2.3.2. Fonction d'adressage

2.3.2.1. *L'adresse internet*

Le module IP distingue deux adresses : l'adresse internet et l'adresse sous-réseau.

L'**adresse internet** a une longueur fixée à 32 bits. C'est elle qui est reprise dans l'en-tête du datagramme internet décrite en annexe D. Cette adresse internet comprend un numéro de réseau et une adresse locale. Le **numéro de réseau** identifie un réseau particulier tandis que l'**adresse locale** identifie un ordinateur hôte particulier d'un réseau. Ceci permet d'identifier les ordinateurs hôtes appartenant à des réseaux différents (RFC 791, p. 24).

Pour répondre à la complexité croissante des structures des réseaux qui apparaissent maintenant à l'intérieur des organisations et pour lutter contre l'accroissement brutal du nombre des numéros de réseau, une nouvelle structure d'adresse internet a été proposée (RFC 950). Dans cette nouvelle structure, on a introduit le concept de sous-réseau. Un **sous-réseau** est un réseau particulier d'une organisation. Pour réaliser cette idée, l'adresse locale a été divisée en deux parties : le numéro de sous-réseau et l'adresse locale. Le numéro de réseau identifie alors l'ensemble des sous-réseaux d'une organisation. Il sera utilisé pour le routage en dehors de l'organisation. Le **numéro de sous-réseau** identifie un sous-réseau de l'organisation. Il est utilisé pour le routage à l'intérieur de l'organisation. L'adresse locale identifie maintenant un ordinateur hôte particulier d'un sous-réseau (RFC 1009, p. 63).

Ces deux types d'adresse internet sont représentés symboliquement de la manière suivante :

adresse internet = { < numéro de réseau >, < adresse locale > }

et

adresse internet = { < numéro de réseau >, < numéro de sous-réseau>,
< adresse locale > }

Différentes classes d'adresse internet ont été prévues pour permettre une adaptation à des configurations de réseaux très différentes. Ces classes sont les suivantes :

- la classe A est prévue pour une configuration comprenant beaucoup d'ordinateurs attachés au(x) réseau(x) de l'organisation;
- la classe C est plus adaptée aux configurations groupant peu d'ordinateurs;
- la classe B représente un compromis entre la classe A et la classe C;
- les classes D et E ont été définies à titre expérimental; nous n'en connaissons pas leur structure.

Le tableau 2.1. résume les différents formats d'adresse internet pour le type d'adresse internet le moins récent (source : RFC 791, p. 24).

Format / Classe	Bits de tête	Numéro de réseau	Adresse locale
A	0	7 bits	24 bits
B	10	14 bits	16 bits
C	110	21 bits	8 bits
D E	Définies à titre expérimentale		

Tab. 2.1. - Format de l'adresse internet

Le choix d'une adresse internet de 32 bits a été justifié au début des années 80 par les concepteurs du protocole IP parce qu'elle permettait de disposer d'un nombre suffisant d'adresses internet pour les développements futurs des catenets sans pour autant atteindre une dimension excessive (Postel 81, p. 261). La rigidité provoquée par la longueur fixe de l'adresse a été nuancée par l'introduction des trois classes d'adresse internet ainsi que par leur structure hiérarchique.

2.3.2.2. *L'adresse sous-réseau*

L'adresse internet est l'adresse reprise dans l'en-tête de chaque datagramme pour en désigner la source et la destination. L'adresse internet destination est exploitée par chaque module IP pour déterminer le chemin à suivre par le datagramme. Pour traverser un réseau particulier, un datagramme doit être encapsulé dans un paquet. Un **paquet** est l'unité de transmission pour un réseau particulier. L'**adresse sous-réseau** est l'adresse reprise dans l'en-tête des paquets.

L'adresse sous-réseau identifie les stations attachées à un réseau particulier. Sa structure est spécifique à chaque réseau. Pour le réseau local Ethernet, cette adresse a 48 bit. L'**adressage** y est **absolu**, c'est-à-dire que l'on ne peut déduire de l'adresse Ethernet, aucune indication sur l'endroit où se trouve la station. Pour le réseau local anneau de Cambridge, l'adresse sous-réseau compte 16 bits. L'adressage y est également absolu. Par contre, l'**adressage** défini par le

CCITT dans l'avis X.121 est **hiérarchique**. Il est structuré de telle sorte que l'on peut retrouver à partir de l'adresse X.121 des indications sur l'endroit où se trouve l'abonné, et que l'analyse des parties qui composent l'adresse X.121 peut être distribuée dans l'espace. Rappelons que l'avis X.121 du CCITT définit le plan de numérotage international pour les réseaux publics de données. L'adresse qui y est définie compte 14 demi-octets. Les trois premiers désignent le pays, le quatrième indique le numéro de réseau à l'intérieur du pays et les dix demi-octets restant identifient l'abonné.

2.3.2.3. *Conversion des adresses locales et des adresses sous-réseaux*

Comme une station attachée à un réseau est identifiée au niveau accès-réseau par son adresse sous-réseau et au niveau inter-réseaux par son adresse internet, une conversion entre l'adresse internet et l'adresse sous-réseau est donc nécessaire. Cette conversion est triviale lorsque les deux adresses adoptent le même format. Elle l'est beaucoup moins lorsque la longueur de l'adresse internet est inférieure à celle de l'adresse sous-réseau.

Pour le réseau local Ethernet, le problème se pose bien évidemment puisque l'adresse Ethernet a une longueur de 48 bits tandis que l'adresse internet n'en compte que 32. La réponse à ce problème est apportée par le protocole ARP (Address Resolution Protocol). La fonction de conversion d'adresses réalisée par ce protocole est typiquement une fonction d'harmonisation de service. Elle est donc du ressort du module IP.

Le protocole ARP, en réalisant cette fonction de conversion, va chercher à lever les deux limites suivantes :

- ne pas seulement être adopté que par le protocole IP, mais l'être également par les autres protocoles de la couche réseau qui coexistent sur le réseau Ethernet; ces protocoles sont, par exemple, le protocole PUP (Parc Universal Packet) de Xerox, le protocole Routing de DEC, le protocole CHAOS, etc. Ces protocoles ont bien évidemment des formats d'adresse différents;

- pouvoir être utilisé sur d'autres types de réseaux tels que, par exemple, les réseaux radio. Présenté de manière théorique, on peut dire que le module ARP cherche à tenir à jour une table de correspondance qui, à partir du triplet

  < type de réseau >, < type de protocole >, < adresse protocole >,

  permet d'obtenir l'adresse sous-réseau souhaitée. Dans le cas qui nous occupe, le type de réseau est le réseau Ethernet, le type de protocole est le protocole IP, l'adresse protocole est l'adresse internet et l'adresse sous-réseau est l'adresse Ethernet.

Pour présenter ce protocole ARP, nous allons d'abord décrire le format du paquet ARP et ensuite exposer de manière intuitive son fonctionnement.

2.3.2.3.1. Format du paquet ARP

Quels que soient le type de réseau, le type de protocole et l'adresse protocole, le champ *données* du paquet aura le format représenté à la figure 2.2.

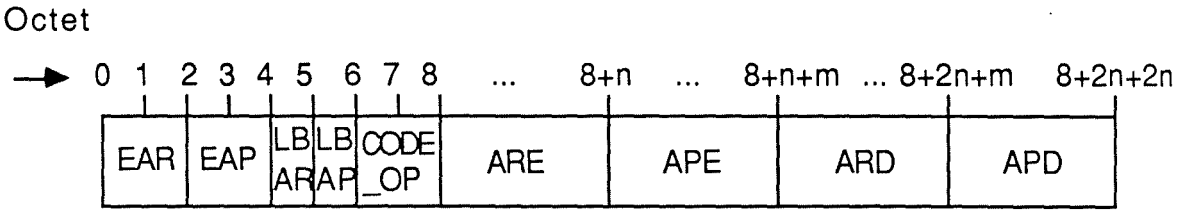


Fig. 2.2. - Format du paquet ARP.

Format :

- le champ EAR (Espace d'Adresse sous-Réseau, 16 bits) identifie le type de réseau utilisé; par exemple, Ethernet ou le réseau radio PRNET;

- le champ EAP (Espace d'Adresse Protocole, 16 bits) identifie le type de protocole inter-réseaux utilisé; par exemple, IP ou PUP;

- le champ LBAR (Longueur en Bytes de l'Adresse sous-Réseau, 8 bits) indique la longueur en bytes des adresses sous-réseau des stations connectées au réseau; par exemple, 6 pour Ethernet;

- le champ LBAP (Longueur en Bytes del'Adresse Protocole, 8 bits) indique la longueur en bytes des adresses du protocole de couche réseau; par exemple, 4 pour le protocole IP;

- le champ CODE-OP (CODE d'OPération, 16 bits) indique s'il s'agit d'une requête ou d'une réponse à une requête; il peut donc prendre les valeurs REQUEST ou REPLY;

- le champ ARE (Adresse sous-Réseau de l'Emetteur, n bytes) indique l'adresse sous-réseau de la station émettrice du paquet; pour le réseau Ethernet, ce champ a une longueur de 6 octets;

- le champ APE (Adresse Protocole de l'Emetteur, m bytes) indique l'adresse du protocole de couche réseau de la station émettrice de paquet; pour le protocole IP, ce champ a une longueur de 4 octets;

- le champ ARD (Adresse sous-Réseau de la Destination, n bytes) indique l'adresse sous-réseau de la station destination du paquet;

- le champ APD (Adresse Protocole de la Destinaiton, m bytes) indique l'adresse du protocole de couche réseau de la station destination du paquet.

Dans le reste de cette présentation du protocole ARP, nous allons nous limiter au type de réseau Ethernet et au protocole IP. Pour être complet, nous devons encore signaler que, sur le câble Ethernet, les paquets Ethernet sont utilisés pour transporter des informations provenant des protocoles IP, PUP, ... et ARP. Pour pouvoir déterminer à quel protocole appartiennent les informations contenues dans les champ données du paquet Ethernet, on attribue une valeur particulière pour chaque protocole au champ *type* du paquet Ethernet. Ainsi, pour le protocole IP, ce champ prendra la valeur 0 X 0800; pour le protocole PUP, ce sera la valeur 0 X 0200 et pour le protocole ARP, ce sera 0 X 0806.

2.3.2.3.2. Protocole ARP

Pour présenter le protocole ARP, nous allons supposer que les machines ALMA et JUNON sont connectées au réseau Ethernet. Leurs adresses Ethernet (AE) sont respectivement AE (ALMA) et AE (JUNON). Supposons encore que le module IP de ALMA veut envoyer un datagramme internet à JUNON. Il va donc demander au module ARP de lui fournir l'adresse AE (JUNON) à partir du type de protocole, IP dans le cas qui nous intéresse, et de l'adresse AI (JUNON). Dans la version 4.2BSD du modèle DoD, le module ARP gère sa table de correspondance ARPTAB de manière dynamique :

- le module ARP n'ajoute une nouvelle entrée dans sa table ARPTAB que si les deux stations ont échangé un datagramme;
- le module ARP supprime de sa table toutes les entrées qui n'ont plus été référencées depuis 20 minutes.

Ce mode de gestion permet de ne conserver dans la table ARPTAB que les stations avec lesquelles des dialogues fréquents ont lieu.

Supposons enfin que dans la table ARPTAB d'ALMA, il n'y a pas (ou plus) d'entrée pour la station JUNON. Pour pouvoir répondre à la requête du module IP, le module ARP va générer le paquet de requête représenté à la figure 2.3.

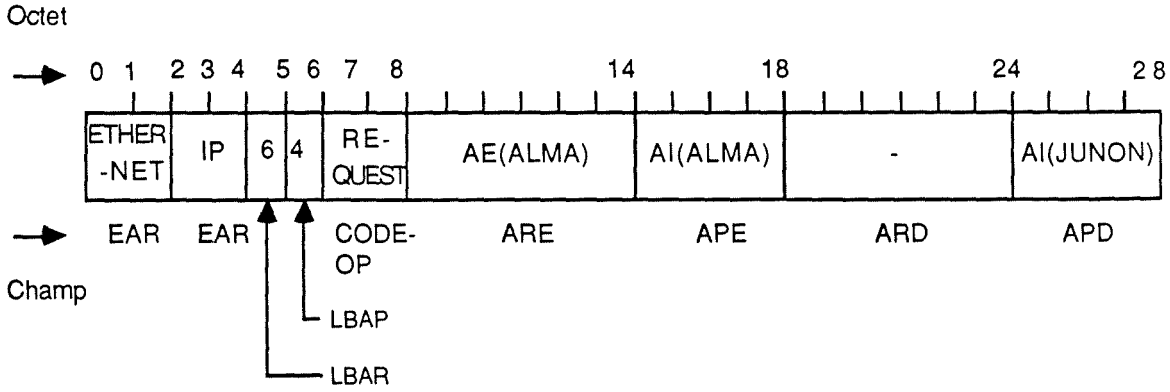


Fig. 2.3. - Exemple de paquet ARP de requête.

Ce paquet ARP de requête va être adressé au module ARP de toutes les stations connectées au réseau de manière à leur permettre de mettre éventuellement à jour leur table de routage avec l'information contenue dans le paquet. Ce paquet ARP sera donc encapsulé dans un paquet Ethernet. Le champ *adresse destination* de ce paquet aura la valeur particulière convenue pour l'adressage en mode diffusion, c'est-à-dire que le champ *adresse destination* sera rempli de bits mis à 1, et le champ *type* prendra la valeur 0 X 0806. Précisons que l'adressage en mode diffusion est utilisé quand une station veut envoyer un paquet à toutes les autres stations raccordées au réseau. Signalons encore que si une station veut envoyer un paquet vers plusieurs autres stations raccordées au réseau, on parlera d'adressage en mode diffusion restreinte.

Quand le paquet arrive à une station, il est décapsulé et renvoyé au module ARP. Nous avons décrit à la figure 2.4. le traitement qu'il y subira. Pour cela, nous avons adopté le formalisme suivant :

- pour désigner l'information contenue dans le paquet, on la postfixera de PAQUET;

- pour désigner une propriété spécifique à la station JUNON, on la postfixera de JUNON. Remarquons bien que les autres stations connectées au réseau reçoivent également le paquet et le traitent.

L'algorithme de la figure 2.4. comprend les six tests suivants :

- le test 1 vérifie que le type de réseau attaché au paquet ARP est celui du réseau auquel la station est connectée; dans la mesure où actuellement le protocole ARP n'est utilisé que sur le réseau Ethernet, ce test n'a aucun sens;

- le test 2 vérifie que le type de protocole attaché au paquet ARP est bien un des protocoles de réseau que la station supporte; ce test n'est important que si différents protocoles de réseau sont utilisés sur le réseau Ethernet;

- le test 3 détermine si la station émettrice du paquet ARP est déjà reprise dans la table ARPTAB de la station; si oui, on met à jour cette table. On remarquera que les tests 1-3 ne dépendent ni du type de paquet ARP (requête ou réponse) ni de la station destination. Ceci signifie qu'ils seront exécutés par chaque station connectée au réseau chaque fois qu'un paquet ARP sera reçu;

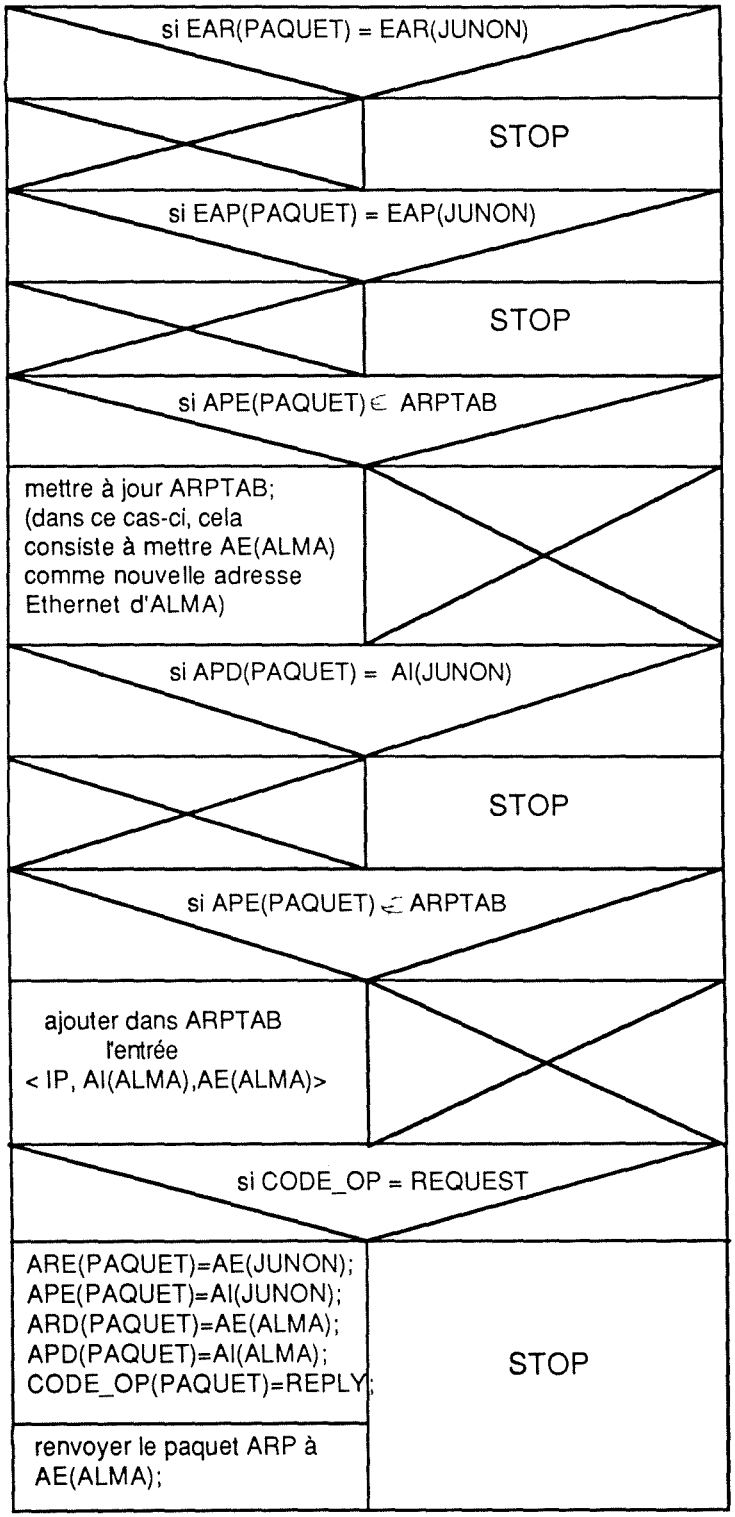


Fig 2.4. - Algorithme de réception d'un paquet ARP.

- le test 4 arrête le traitement du paquet si la station n'est pas la station destination du paquet;

- avec le test 5, on vérifie qu'on dispose bien d'une entrée dans la table ARPTAB pour la station émettrice du paquet ARP; sinon, on ajoute dans la table une entrée pour cette station; rappelons que la décision de mettre à jour la table n'est prise que si les deux stations ont échangé un datagramme;

- si le type de paquet est une requête, on renvoit un paquet ARP de réponse à la station émettrice. Ce paquet de réponse est décrit à la figure 2.5.

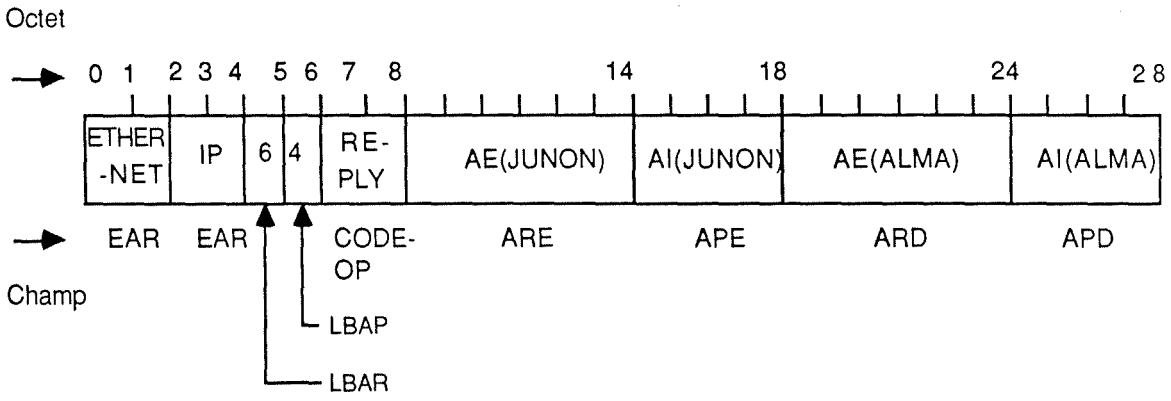


Fig. 2.5. - Exemple de paquet ARP de réponse.

2.3.2.4. *Les adresses OSI*

Dans le modèle OSI comme dans le modèle DoD, on distingue à la couche réseau différentes adresses. Pour le modèle OSI, il s'agit de l'adresse sous-réseau, de l'adresse du point d'accès au service de réseau (adresse NSAP, pour Network Service Access Point) et des informations d'adresse du protocole de réseau (NPAI, pour Network Protocol Address Information). Nous allons d'abord énoncer les définitions de ces trois adresses. Ensuite, nous présenterons la norme ISO/DIS 8348/ADD2 qui définit l'adressage OSI dans la couche réseau. Ceci nous permettra de présenter les formats de l'adresse NSAP et des NPAI. Enfin, on énoncera les règles qui ont guidé la conception de ces adresses.

2.3.2.4.1. Définitions

L'adresse sous-réseau est l'information utilisée dans le contexte d'un sous-réseau réel particulier pour identifier un point d'attache à ce sous-réseau. Un **point d'attache** à un sous-réseau est le point où un système réel extrémité, une passerelle ou un autre sous-réseau réel est relié à ce sous-réseau réel (ISO/DIS 8348/ADD2, p. 6). La figure 2.6. représente ce concept de point d'attache à un sous-réseau (source : ISO/DIS 8348/ADD2, p.6).

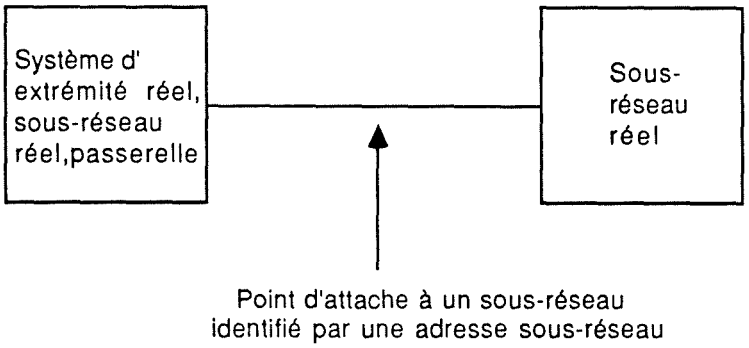


Fig. 2.6. - Point d'attache à un sous-réseau réel.

Pour un réseau X.25, cette adresse sous-réseau est celle qui a été spécifiée par l'avis X.121 du CCITT. Cette adresse est aussi appelée dans la littérature, adresse ETTD. En fait, cette adresse sous-réseau est celle utilisée à l'intérieur du sous-réseau pour le routage des paquets.

L'adresse du point d'accès au service est l'information dont le fournisseur du service de réseau OSI a besoin pour identifier un point d'accès au service réseau particulier. **L'information d'adresse du protocole de réseau** est l'information encodée dans une NPDU (Network Paquet Data Unit) pour conserver la sémantique d'une adresse NSAP. La différence entre ces deux dernières adresses est la suivante : l'adresse NSAP est utilisée lors du passage des paramètres d'adresse source et d'adresse destination avec les primitives de service de réseau tandis que les NPAI sont les informations qui sont contenues dans l'en-tête des NPDU et qui sont relatives aux adresses source et destination. Il est clair qu'il existe une relation entre les adresses NSAP qui apparaissent dans les primitives de service réseau et les NPAI qui apparaissent dans les en-têtes des NPDU dans la mesure où la sémantique des adresses NSAP doit être conservée dans les NPAI (ISO/DIS 8348/ADD2, p. 8). Ces deux adresses sont associées à la sous-couche SNICP de la couche réseau tandis que l'adresse sous-réseau est associée à la sous-couche SNACP.

2.3.2.4.2. Format de l'adresse NSAP

Nous allons présenter maintenant le format de l'adresse NSAP. Chaque adresse NSAP comprend au moins trois parties qui sont l'identificateur de l'autorité et du format (AFI, pour Authority and Format Identifier), l'identificateur du domaine initial (IDI, pour Initial Domain Identifier) et la partie spécifique au domaine (DSP, pour Domain Specific Part). Cette adressage est hiérarchique dans la mesure où la signification de chaque partie dépend de la valeur des parties précédentes.

L'identificateur AFI (entier compris entre 0 et 99) indique le format des deux autres champs et spécifie l'autorité responsable de l'allocation des valeurs du champ IDI. Par exemple, la valeur 36 du champ AFI signifie que les deux parties restantes sont composées de chiffres décimaux, que l'identificateur IDI est une adresse X.121 et donc que l'autorité qui assigne les valeurs au champ IDI est celle qui attribue les adresses X.121. Pour la Belgique, il s'agit de la RTT.

D'une manière générale, l'identificateur AFI désigne un domaine d'adresses réseau. Ce domaine est constitué de toutes les adresses NSAP attribuées par une ou plusieurs autorités. Toutefois, toute adresse NSAP d'un domaine d'adresses réseau n'est administrée que par une seule autorité d'adressage. Les domaines d'adresses actuellement possibles sont les suivants :

- les réseaux publics : pour les réseaux à communication par paquets, AFI = 36; pour les réseaux télex, AFI = 40; pour les réseaux téléphoniques, AFI = 42; pour les réseaux numériques à intégration de services, AFI = 44;

- un pays ou une région : dans ce cas, la valeur de AFI est 38; le champ IDI identifie le pays ou la région; pour la Belgique, sa valeur est 206 et l'autorité responsable de l'allocation des valeurs attribuées au champ DSP est l'Institut Belge de Normalisation; le champ IDI sera dans ce cas un entier compris entre 0 et 999 qui formera le Data Country Code (DCC);

- une organisation internationale : dans ce cas, la valeur de AFI est 34; le champ IDI identifie l'organisation responsable de l'allocation des valeurs attribuées au champ DSP; il a une valeur comprise entre 0 et 9999. La sémantique et le format du champ DSP sont déterminés par l'autorité identifiée par le champ IDI. La figure 2.7. représente l'adresse NSAP définie par le projet ISO/DIS 8348/DAD2.

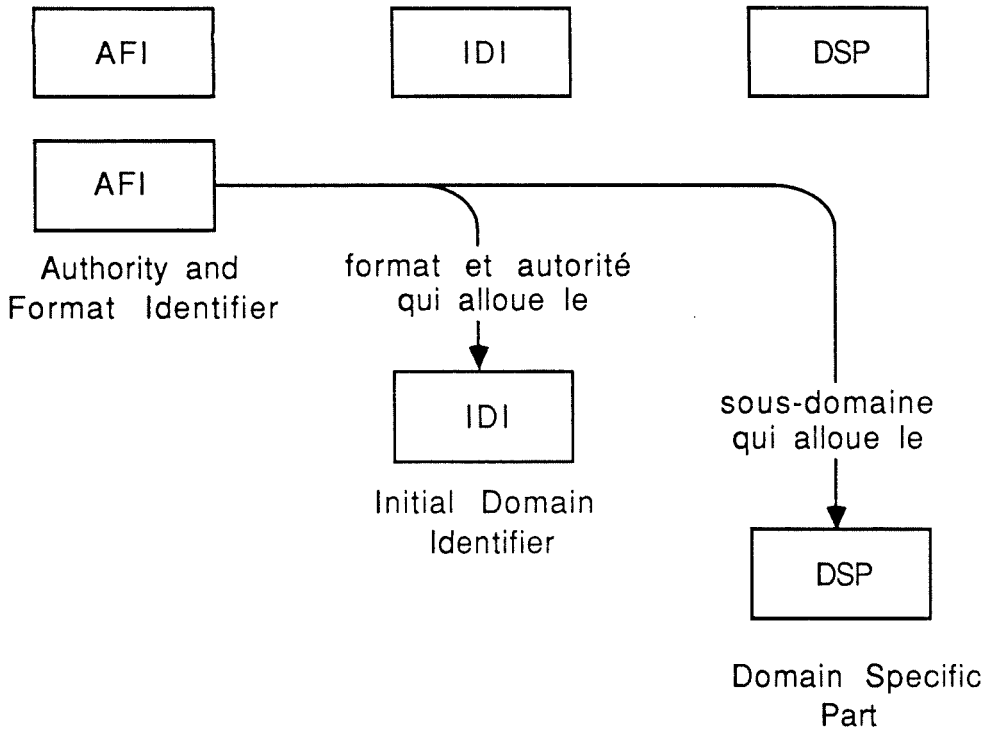


Fig. 2.7. - Structure de l'adresse NSAP.

2.3.2.4.3. Principes de conception

Lors de la conception de l'adresse NSAP et du NPAI, l'ISO a essayé de rencontrer les objectifs suivants :

- la structure des adresses NSAP doit être hiérarchique;
- chaque adresse NSAP doit être un identifiant global; elle doit pouvoir être utilisée partout et identifier toujours le même point d'accès au service réseau;
- l'utilisateur du service réseau ne peut influencer avec l'adresse NSAP le chemin qui sera utilisé par le fournisseur du service réseau;
- l'adresse NSAP ne pourra contenir aucune indication explicite sur le type de service (avec ou sans connexion) ou la qualité de service (débit, temps de transit, taux d'erreur résiduel,...) demandés par l'utilisateur de ce service (ISO/DIS 8348/ADD2, pp. 10-13).

2.3.3. Fonction de routage

2.3.3.1. *Introduction*

Etroitement lié à la fonction d'adressage, le problème du routage des datagrammes internet concerne le choix du chemin que ces datagrammes doivent emprunter. Pour le module IP d'un ordinateur hôte source ou d'une passerelle, il s'agit en particulier de déterminer à quelle passerelle ou à quel ordinateur hôte du catenet, le datagramme doit être transmis. Le protocole IP distingue différents types de routage en fonction du degré de liberté laissée au module IP des passerelles de choisir ce chemin.

Dans le cas le plus fréquent où les passerelles sont libres de déterminer le chemin des datagrammes, leur module IP consulte une **table de routage** qui donne, pour chaque réseau destination possible, la prochaine passerelle à qui le datagramme doit être envoyé (Stallings 85, p. 455). La manière dont cette table est gérée ne relève pas du protocole IP; elle dépend pour l'essentiel de l'implémentation locale du module IP.

Le problème du routage présente deux aspects importants : l'aspect de décision et l'aspect de mise à jour. La décision concerne la façon dont les itinéraires des datagrammes sont choisis. Cette décision dépend du type de routage mais elle doit également intégrer un certain nombre d'éléments dont ceux qui concernent la sécurité, la priorité et la qualité de service ne sont pas les moins importants. L'aspect mise à jour concerne le problème de la reconfiguration du catenet en cas de panne, d'anomalie ou de congestion.

Pour étudier la fonction de routage, nous allons procéder de la manière suivante :

- au point 2.3.3.2., nous présenterons les trois types de routage disponibles avec le protocole IP;
- au point 2.3.3.3., nous décrirons comment l'aspect décision du routage est réalisée dans la version 4.2BSD du protocole IP;
- au point 2.3.3.4., nous dresserons la liste des différents modes de gestion des tables de routage;
- au point 2.3.3.5., nous présenterons le protocole utilisé pour mettre à jour les tables de routage pour la version 4.2BSD du protocole IP.

Rappelons que le paragraphe 2.3.4. est entièrement consacré aux passerelles dont le rôle dans la fonction de routage est essentiel.

Dans le projet ISO/DIS 8473, on considère la fonction de routage comme l'un des éléments les plus déterminants de la gestion de la couche de réseau. Elle constitue avec le contrôle de réseau l'un des aspects majeurs du **système de gestion de réseau**. Ce système de gestion de réseau est en cours de spécification. Il remplira normalement les fonctions suivantes :

- le routage,
- l'analyse statistique,
- le contrôle de réseau,
- la mise en correspondance de répertoire,
- le contrôle d'encombrement,
- le contrôle de configuration,
- la comptabilité.

Ceci étant signalé, un certain nombre de mécanismes ou de contraintes concernant la fonction de routage a été prévu dans le protocole ISO 8473; ils seront mentionnés au moment opportun dans ce paragraphe.

2.3.3.2. *Types de routage disponibles avec le protocole IP*

L'acheminement d'un datagramme peut parfois ne comprendre qu'une seule étape. C'est le cas lorsque les ordinateurs hôtes source et destination sont connectés au même réseau. A d'autres moments, quand ils sont reliés à des réseaux différents, cet acheminement peut exiger différentes étapes. Dans le premier cas, la décision de routage est évidente même si celle-ci peut entraîner des mécanismes complèxes de routage au niveau inférieur. Dans le second cas, cette décision nécessite de définir une série de passerelles du catenet par lesquelles le datagramme doit transiter.

Pour cataloguer les différents types de routage, un élément à prendre particulièrement en considération est l'endroit où une décision de routage est prise. Si l'ordinateur hôte source doit spécifier l'ordinateur hôte destination ainsi que toutes les passerelles par lesquelles le datagramme doit transiter, on parlera de **routage source**. Si, par contre, l'ordinateur hôte source doit seulement spécifier l'ordinateur hôte destination et la première passerelle à qui adresser le datagramme et si chaque passerelle doit choisir à qui le transmettre, on parlera alors de **routage incrémental**. Pour le premier type de routage, le rôle de la passerelle se limite à la fonction de relais tandis que dans le second, s'ajoute la fonction de routage.

Le protocole IP tout comme le protocole ISO 8473 proposent ces deux types de routage aux utilisateurs de leur service. Pour les deux protocoles, le routage sera par défaut incrémental tandis que le routage source sera disponible en option.

Deux sortes de routage source sont disponibles en option dans les deux protocoles : il s'agit du routage source lâche et du routage source strict. Le **routage source lâche** signifie que l'émetteur du datagramme ne veut spécifier que certaines étapes de l'itinéraire que le datagramme doit suivre. Entre deux passerelles mentionnées, le datagramme peut emprunter différents itinéraires. On appelle aussi cette forme de routage source, routage hybride parce qu'il présente également des aspects du routage incrémental. Le **routage source strict** oblige l'émetteur du datagramme à mentionner toutes les passerelles par lesquelles le datagramme devra transiter pour arriver à l'ordinateur hôte destination.

Pour savoir lequel de ces trois types de routage une passerelle doit appliquer, celle-ci doit analyser l'en-tête du datagramme reçu; si l'en-tête du datagramme contient un champ d'option de routage source strict ou de routage source lâche, c'est ce type de routage qui sera réalisé; sinon, on appliquera le routage incrémental. Dans le champ d'option, on retrouvera la liste des adresses internet des passerelles à visiter.

Le choix de fournir une option de routage source est justifié par les concepteurs du modèle DoD par le fait que plusieurs petits réseaux peuvent avoir été connectés à un catenet par des arrangements ad hoc. Dès lors, leur numéro de réseau ne sont pas toujours connus des passerelles (Postel 81, pp. 269-270).

Pour être complet, signalons encore que les protocoles IP et ISO 8473 ajoutent un autre mécanisme en relation avec la fonction de routage; il s'agit de l'enregistrement de la route. Ce mécanisme disponible en option fournit un moyen d'enregistrer le chemin que suit un datagramme par l'insertion, à chaque passerelle visitée par le datagramme, de l'adresse internet de la passerelle dans le champ d'enregistrement de la route du datagramme.

Dans le reste de ce paragraphe, nous allons décrire comment le routage incréméntal est réalisé dans la version 4.2BSD du modèle DoD.

2.3.3.3. *Description de l'implémentation du routage incrémental pour la version 4.2BSD du modèle DoD*

Idéalement, la fonction de routage d'un protocole de niveau inter-réseaux devrait intégrer les aspects suivants :

- les types de routage (incrémental, source, hybride),
- la sécurité, la priorité,
- les qualités de service.

C'est bien ce qui a été prévu dans le protocole ISO 8473 (ISO/DIS 8473, p. 18, p. 47).

En ce qui concerne le protocole IP, aucune contrainte concernant la sécurité, la priorité et les qualités de service n'a été spécifiée. Il faut bien reconnaître que ces éléments relèvent le plus souvent de l'implémentation locale surtout en ce qui concerne l'élément de sécurité. Les rares documents (Stallings 85, p. 456; Hinden 83) que nous avons trouvés abordant ce sujet, ne présentent des tables de routage que sous la forme d'un tableau qui, à chaque numéro de réseau fait correspondre l'adresse internet de la passerelle à qui envoyer le datagramme. Nous avons voulu vérifier si, pour la version 4.2BSD du modèle DoD, cette simplicité était aussi de mise.

Dans la version 4.2BSD, nous avons pu constater que chaque module IP dispose de deux tables de routage dont les éléments sont des chemins. Dans la première table, on retrouve une liste de chemins dont la destination est un ordinateur hôte. Dans la seconde table, on retrouve une liste de chemins qui ont pour destination un réseau. Chaque chemin de ces tables est décrit par la structure *rtentry*. Ces tables sont organisées comme des suites chaînées d'éléments de structure *rtentry*. Pour un ordinateur hôte ou un réseau destination donné, on peut trouver zéro, un ou plusieurs éléments qui décrivent le chemin pour y arriver.

Cette structure *rtentry* a la forme suivante :

```
struct rtentry {
  u_long  rt_hash;
  struct  sockaddr rt_dst;
  struct  sockaddr rt_gateway;
  short  rt_flags;
  short  rt_refent;
  u_long rt_use;
  struct  ifnet *rt_ifp;
  };
```

Pour retrouver le(s) élément(s) qui correspond(ent) à un ordinateur hôte ou à un réseau destination particulier dans leur table, on calcule une clé de 32 bits sur base de l'adresse internet de l'ordinateur hôte destination ou du numéro du réseau destination, suivant le cas. Tous les éléments *rtentry* dans les tables, qui ont un champ *rt_hash* dont la valeur est égale à la clé calculée, sont des éléments qui correspondent à l'ordinateur hôte ou au réseau destination souhaité. Le champ *rt_hash* est donc une clé de hashing qui est utilisée pour accélérer les recherches dans les tables de routage.

Chaque entrée dans les tables de routage contient une structure[3] de données *rt_dst* qui décrit l'adresse de la destination et une autre *rt_gateway* qui décrit l'adresse de la prochaine passerelle par où le chemin passe. Si la destination du chemin est un réseau, alors la partie de *rt_dst* qui contient l'adresse de l'ordinateur hôte prend la valeur nulle. Le champ *rt_flags* est utilisé pour indiquer

- -si le chemin est utilisable;
- -si la destination est un ordinateur hôte ou un réseau;
- -si la destination et la station locale sont connextées au même réseau.

---

[3] Cfr. paragraphe 6.3.2 pour la description de structure *sockaddr*.

Le champ *rt_refent* compte le nombre de références à la route tandis que *rt_use* compte le nombre de datagrammes qui ont été envoyés sur cette route. *rt_ifp* décrit l'interface réseau qui sera utilisée pour envoyer le datagramme par cette route. Ceci est très utile si la station est connectée à plusieurs réseaux.

Pour trouver le chemin vers une destination particulière, on balaie en premier lieu la table de routage des chemins vers les ordinateurs hôtes; si aucune entrée n'y a été trouvée, on balaie ensuite la table de routage des chemins vers les réseaux. Si ces deux recherches n'ont abouti à aucun résultat, il existe également dans la table de routage des réseaux un enregistrement dont le champ destination ne contient que des zéros. Cet enregistrement désigne le chemin qui doit être utilisé dans cette situation. Il représente en quelque sorte le chemin par défaut. Avec cette solution, on espère bien que la station ainsi désignée dispose de l'information nécessaire pour trouver un chemin vers cette destination. A l'opposé, si plusieurs chemins sont disponibles pour arriver à la destination, on choisira le chemin qui a le taux d'utilisation *rt_use* le plus bas. Cette solution a été conçue de manière à éviter les problèmes de congestion[4].

Le but que nous avons poursuivi avec cette description était de voir quels éléments allaient être pris en considération dans la version 4.2BSD lors du choix du chemin des datagrammes quand le routage est incrémental. Nous avons pu constater que le seul mécanisme introduit cherche à éviter les situations de congestion. Les autres éléments tels que la sécurité, la priorité et la qualité de service ont été laissés de côté.

2.3.3.4. *Gestion des tables de routage*

Au point 2.3.3.2., nous avions distingué différents types de routage en ayant pris en considération l'endroit où une décision de routage tombait. Ici, nous allons définir de nouvelles sortes de routage en tenant compte de deux autres éléments qui sont relatifs à la gestion des tables de routage; il s'agit

- de la fréquence des mises à jour des tables de routage et
- du mécanisme de contrôle des modifications de ces tables.

En tenant compte de la fréquence des remises à jour des tables, on distingue deux nouveaux types de routage : le routage fixe et le routage adaptatif. "*On entend par **routage fixe** une méthode de routage dans laquelle les tables de routage ne sont que rarement remises à jour, par opposition au **routage adaptatif**, avec lequel les tables de routage sont corrigées en permanence, en fonction par exemple de l'état instantané du trafic. On voit que la méthode du routage fixe vise une optimisation globale et à long terme, alors que la méthode adaptative a pour ambition de satisfaire à tout instant au critère d'optimalité*" (Nussbaumer 87, p. 101).

---

[4] Le problème de congestion est présenté au paragraphe 4.3.1.

Le routage fixe est bien adapté au réseau dont le topologie est stable et le trafic moyen. Le routage adaptatif est très approprié pour les réseaux dont la configuration change fréquemment et pour lesquels la probabilité de panne, d'anomalie ou de congestion est plus grande.

Pour ce dernier type de routage, on en distingue encore trois autres en fonction de leur mécanisme de contrôle des modifications. "*Si les remises à jour ne peuvent être apportées que par une autorité centrale, le* ***routage adaptatif*** *sera* ***centralisé****. Si chaque passerelle essaie de tenir à jour ses tables de routage en essayant périodiquement des chemins alternatifs ou en observant les performances de manière isolée, on a affaire à un* ***routage adaptatif isolé****. Si, par contre, les passerelles s'échangent périodiquement des informations de routage, il s'agit là de* ***routage adaptatif distribué*** " (Shoch 78, p. 280).

2.3.3.5. *Le protocole RIP*

Dans la version 4.2BSD, le routage peut être adaptatif centralisé ou adaptatif distribué. Le programme, appelé **ROUTE**, permet de modifier directement les tables de routage par l'insertion d'un nouveau chemin, la modification d'un chemin existant ou la supression d'un chemin. Ce programme manipule en quelque sorte manuellement les tables de routage. Il ne peut être exécuté que par le super-utilisateur. Ceci donne un caractère centralisé à ce routage. Cette solution est appropriée pour des configurations stables et peu importantes.

Dans la plupart des cas, le routage sera adaptatif distribué. Ainsi, au moment du reboot du système, un processus daemon, appelé **ROUTED**, sera créé. Ce processus va créer et gérer les tables de routage de manière adaptative et distribuée. Pour ce faire, il implémente une variante du protocole RIP (Routing Information Protocol) de XNS (Xerox Network System).

Nous allons décrire le protocole RIP en procédant de la même manière que pour le protocole ARP, c'est-à-dire en présentant d'abord le format du paquet RIP et ensuite le fonctionnement du protocole. Nous n'avons pas eu à notre disposition le document qui décrit la version 4.2BSD du protocole RIP. La description qui suit, provient du manuel de commandes UNIX pour la commande *routed*, de l'article de base de Xerox (XNS) qui décrit le protocole RIP et du code source de la commande *routed*.

2.3.3.5.1. Format du paquet RIP

Le protocole RIP utilise le service datagramme offert par le protocole UDP pour échanger ses paquets. Ceci signifie que chaque paquet RIP est encapsulé dans un datagramme UDP avant d'être envoyé. Le format du paquet RIP de la version 4.2BSD diffère légèrement de celui de la version de XNS représenté à la figure 2.8. Nous ne sommes toutefois pas en mesure de présenter ici le format précis du paquet RIP pour la version 4.2BSD.

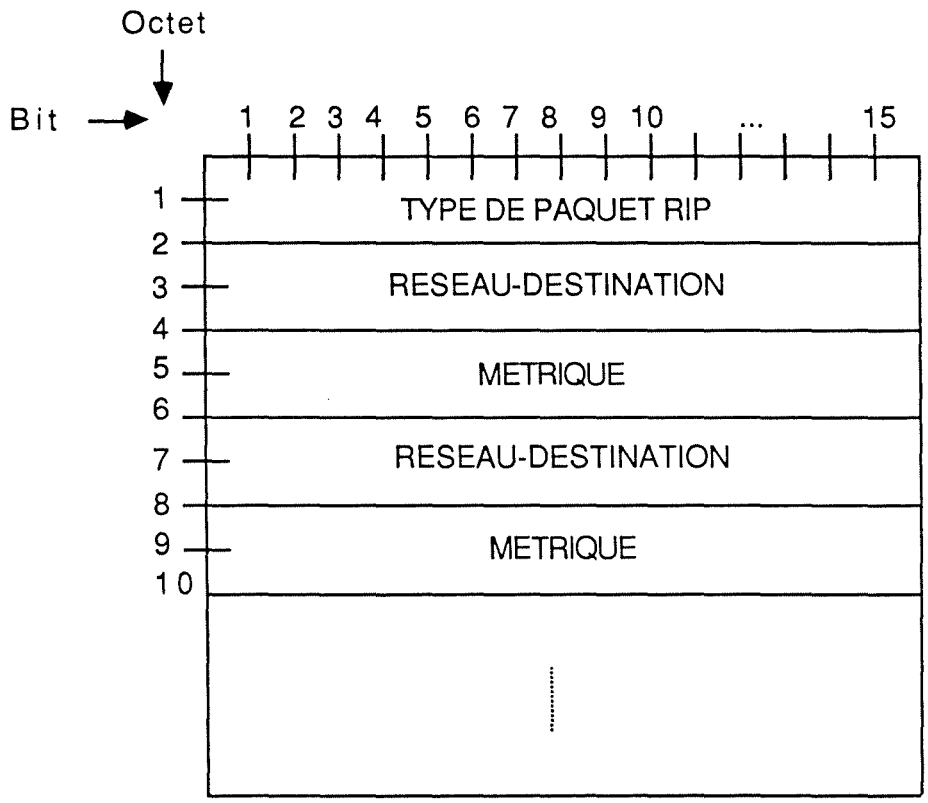


Fig. 2.8. - Format du paquet RIP pour la version XNS.

Format :

- le champ *type de paquet RIP* (16 bits) indique qu'il s'agit d'une requête ou d'une réponse à une requête; il peut donc prendre les valeurs RIPCMD-REQUEST et RIPCMD-RESPONSE;

- les deux champs qui suivent, *réseau-destination* (32 bits) et *métrique* (16 bits) peuvent apparaître une ou plusieurs fois consécutivement. Leur signification change suivant le type du paquet RIP.

Si le *type de paquet RIP* est une requête, alors, dans chaque doublet, le champ *réseau-destination* désigne le réseau qui fait l'objet d'une demande de renseignement et le champ *métrique* prend une valeur infinie (en pratique, fixée à 16). Le champ *métrique* indique le nombre de passerelles qu'un datagramme internet devra traverser avant d'arriver à destination. Dans le cas d'une requête, ce nombre n'est pas encore connu; ce qui explique que le champ prend une valeur infinie. Si le *type de paquet RIP* est une réponse, alors, dans chaque doublet, le champ métrique indiquera le nombre de passerelles qu'il faudra traverser avant d'arriver au réseau décrit par le champ *réseau-destination* en partant de la passerelle qui a généré le paquet de réponse. Un paquet RIP de réponse avec un champ *métrique* prenant une valeur infinie signifie que le réseau destination est hors de portée ou que la passerelle ne le connait pas.

2.3.3.5.2. Protocole RIP

Le protocole RIP est utilisé par chaque passerelle pour tenir à jour leur table de routage. La présence du protocole RIP sur les ordinateurs hôtes s'explique par le fait que la version 4.2BSD du modèle DoD permet à un ordinateur hôte de jouer le rôle d'une passerelle s'il possède des interfaces sur différents réseaux.

Le protocole RIP distingue deux sortes de passerelles : les passerelles normales et les passerelles inter-réseaux. La **passerelle normale** n'utilise le protocole RIP que pour demander des renseignements et ainsi tenir à jour ses propres tables de routage. La **passerelle inter-réseaux** intervient pour demander et surtout fournir des renseignements. Cette sorte de passerelle passe une partie de son temps à échanger des informations de routage de manière à donner aux passerelles normales une image fidèle de la situation du catenet. Il doit y avoir au moins une passerelle inter-réseaux attachée à chaque réseau du catenet.

Le protocole RIP est utilisé pour rendre rapide l'initialisation des tables de routage au moment de la remise en marche des ordinateurs hôtes et des passerelles; il permet également une remise à jour rapide des tables suite à des modifications apparues dans la configuration du catenet.

Chaque passerelle qui implémente le protocole RIP dispose de tables de routage légèrement différentes de celles décrites au point 2.3.3.3. A chaque entrée dans les tables, trois nouveaux attributs *rt_metric*, *rt_timer* et *rt_router* ont été ajoutés. L'attribut *rt_metric* a la même signification que le champ *métrique* du paquet RIP. Il mesure le nombre de passerelles à traverser pour arriver au réseau ou à l'ordinateur hôte destination. Ce nombre fournit une estimation de la distance entre la passerelle et la destination. L'attribut *rt_timer* est utilisé pour tenir à jour les tables. Chaque fois qu'un chemin est mis à jour, son attribut *rt_timer* est mis à zéro. Si le chemin n'a plus été mis à jour depuis plus de 3 minutes, le chemin est indiqué non valide. Une minute plus tard, il sera supprimé de la table. L'attribut *rt_router* est une structure de données qui décrit la passerelle inter-réseaux qui est responsable de signaler à la passerelle locale toute modification apportée au chemin en question.

Quand le programme *routed* commence son exécution, il compte d'abord le nombre d'interfaces réseaux actifs présents sur le système. S'il en compte plus d'un, cela signifie que l'ordinateur hôte joue également le rôle de passerelle entre les réseaux. Si l'exécution du daemon *routed* a été lancée avec l'option -s, la passerelle sera de type inter-réseaux. Sinon, ce sera une passerelle normale. Ensuite, *routed* va envoyer en mode diffusion si c'est possible, un paquet RIP de requête sur chaque réseau, puis rentrer dans une boucle pour traiter les paquets RIP de requête et de réponse.

Les paquets RIP qui arrivent sont traités de la manière suivante :

- quand il s'agit d'un paquet de requête, que la passerelle soit normale ou de type inter-réseaux, *routed* va formuler une réponse sur base des renseignements dont il dispose dans ses propres tables. Chaque paquet de réponse généré contient la liste des chemins connus de la passerelle pour chaque réseau destination souhaité. Chacun des champs *métrique* est mesuré par rapport à la passerelle. L'adresse de la passerelle sera mentionnée dans le champ d'adresse source du datagramme IP qui transporte le paquet RIP;

- quand il s'agit d'un paquet de réponse, *routed* ne mettra à jour ses propres tables de routage que si une des conditions suivantes est satisfaite :

(1) - il n'existe encore aucun chemin dans les tables de routage qui a la destination contenue dans le paquet et le champ *métrique* indique que cette destination n'est pas hors de portée;

(2) - le chemin décrit dans le paquet est déjà repris dans les tables et l'émetteur de ce paquet est également la passerelle inter-réseaux désignée par *rt_router* qui est attachée à ce chemin;

(3) - il existe déjà un chemin dans les tables de routage qui a la même destination que celle contenue dans le paquet, mais le chemin décrit dans le paquet est plus court que celui repris dans les tables;

(4) - il existe déjà dans les tables de routage un chemin qui a la même destination et le même métrique que ceux contenus dans le paquet, mais le chemin décrit dans les tables n'a plus été mis à jour depuis au moins 90 secondes.

Dans le premier cas, *routed* va ajouter une nouvelle entrée dans les tables de routage. Dans les deux derniers cas, on va remplacer le chemin existant par celui décrit dans le paquet. Dans tous les cas, l'attribut *rt_timer* sera initialisé à zéro et un paquet de réponse sera généré et adressé à toutes les passerelles qui sont connectés aux réseaux que la passerelle relie.

En plus du traitement des paquets de requête et de réponse que nous venons de présenter, *routed* vérifie toutes les 30 secondes chaque chemin repris dans ses tables de routage. Si un chemin n'a plus été mis à jour depuis 180 secondes, son attribut *rt_metric* est mis à l'infini. Ceci signifie qu'il est marqué pour être détruit. Une minute plus tard, il sera effectivement détruit. Ce délai d'une minute est laissé aux autres ordinateurs du catenet pour constater également que le réseau ou l'ordinateur hôte est hors de portée.

Si la passerelle est de type inter-réseaux, elle transmettra toutes les 30 secondes une copie de ses tables de routage (en un ou plusieurs paquets de réponse) à toutes les passerelles inter-réseaux situées sur tous les réseaux auxquels elle est directement connectée. Si l'adressage en mode diffusion existe pour l'un ou l'autre de ces réseaux, tous les ordinateurs hôtes et toutes les passerelles normales pourront également recevoir ces tables. De plus, chaque

fois qu'une passerelle inter-réseaux modifie l'attribut *rt_metric* d'un des chemins repris dans ses tables, elle est tenue d'en informer immédiatement toutes les autres passerelles inter-réseaux qui lui sont directement connectées avec un paquet de réponse. Ceci permet de diffuser très rapidement à travers tout le catenet les changements survenus dans les informations de routage. Si la passerelle est sur le point de cesser de fonctionner, elle enverra un paquet de réponse indiquant que tous les réseaux destination sont hors de portée si le chemin passe par elle. Ceci permet d'informer le catenet que la passerelle est inutilisable avant qu'elle ne le soit effectivement.

Quand une passerelle inter-réseaux reçoit un paquet de requête, elle essayera de fournir les renseignements souhaités. Si elle ne dispose pas d'une entrée dans ses tables pour le réseau destination souhaité, elle renverra un paquet de réponse avec un champ *métrique* de valeur infinie pour signaler que le chemin est hors de portée.

Mis en oeuvre, le protocole RIP se comporte bien si la configuration du catenet ne change pas trop fréquemment. Que se passe-t-il si une nouvelle passerelle entre en fonctionnement?

Supposons que cette nouvelle passerelle rend le chemin vers certains réseaux plus courts que ceux qui existaient déjà. Toutes les tables de routage du catenet auront été mis à jour pour reflèter la nouvelle configuration dans un délai approximatif de 30 * max (n,15) secondes où n est le nombre de passerelles entre la nouvelle passerelle et le réseau le plus éloigné et les 30 secondes est l'interval de temps entre deux transmissions de copies de tables de routage par une passerelle inter-réseaux.

Si une passerelle inter-réseaux devient inactive et s'il existe d'autres chemins sans doute plus longs pour arriver à certains réseaux destination, la situation sera rétablie dans un délai d'environ 90 secondes. Ce délai représente le temps écoulé après lequel une entrée qui n'a plus été mis à jour devient suspecte et est susceptible d'être remplacée par un chemin moins favorable.

Si un réseau devient inaccessible, l'entrée pour ce réseau sera éliminée de toutes les tables de routage dans un délai approximatif de 90 + 30 * max (15,n).

Les trois comportements que nous venons de décrire, peuvent être améliorés si l'on tient compte que chaque passerelle inter-réseaux doit immédiatement envoyer un paquet de réponse chaque fois qu'elle modifie l'attribut *rt_metric* d'un des chemins repris dans ses tables.

De la description de la mise en oeuvre du protocole RIP que nous venons de faire, on peut déduire que le protocole RIP est approprié pour de grosses configurations de réseaux relativement stables. Il serait moins adéquat pour des configurations changeant relativement fréquemment.

2.3.4. Les passerelles

2.3.4.1. *Introduction*

Nous pouvons résumer tout ce qui a déjà été dit sur les passerelles en rappelant que la **passerelle** est le moyen utilisé dans le modèle DoD pour fournir des chemins d'accès entre les réseaux de telle manière qu'un ordinateur hôte attaché à un réseau puisse communiquer avec un ordinateur hôte relié à un autre réseau. La passerelle joue un rôle important dans le **routage** des datagrammes internet.

Une autre fonction essentielle remplie par les passerelles est celle de **relais** entre les réseaux. Cela signifie qu'une passerelle doit pouvoir recevoir un paquet provenant d'un réseau, le décapsuler pour en retirer le datagramme et, après le traitement de celui-ci par le module IP, le récapsuler dans un nouveau paquet pour l'envoyer ensuite sur un autre réseau. Cette fonction de relais est étroitement lié au problème d'interfaçage des passerelles. Ce problème se pose également à tout ordinateur hôte qui inclut un module IP.

En plus de ces deux fonctions, chaque passerelle réalise aussi les autres fonctions attachées au protocole IP. Elle implémente également une série de protocoles (ICMP, RIP, ...) qui contrôlent et gèrent le catenet. Pour assurer toutes ces fonctions, différents types de passerelles existent. Nous avons déjà mentionné, lors de la description du protocole RIP, les passerelles normales et les passerelles inter-réseaux. Il en existe d'autres : par exemple, celles qui implémentent, comme dans la version 4.2BSD, tous les protocoles du niveau accès-réseau au niveau applications.

La suite de ce paragraphe va être organisée de la manière suivante : au point 2.3.4.2., nous présenterons les problèmes liés à l'interfaçage des réseaux; au point 2.3.4.3., nous décrirons les différents protocoles qu'une passerelle doit implémenter pour remplire sa mission; au point 2.3.4.4., nous présenterons différents types de passerelles.

Le document qui a servi de base à la rédaction de ce paragraphe est (RFC 1009).

2.3.4.2. *Problèmes liés à l'interfaçage des réseaux*

Pour chaque réseau connecté, chaque passerelle comme chaque ordinateur hôte doivent implémenter une série de fonctions qui permettent d'adapter les services offerts par le réseau aux services demandés par le protocole IP. Le protocole IP demande à chaque réseau connecté au catenet qu'il fournisse

un service datagramme. Un problème d'harmonisation de services se pose quand un réseau offre un autre service que le service datagramme.

C'est le cas avec le réseau X.25 qui offre un service circuit virtuel. A côté de ce problème d'harmonisation des services, le problème d'interfaçage présente quatre autres aspects :

- l'encapsulation et la décapsulation des datagrammes internet; pour traverser un réseau particulier, un datagramme va d'abord être encapsulé dans un paquet de ce réseau; arrivé à l'extrémité de ce réseau, ce datagramme sera décapsulé pour être, si cela est nécessaire, à nouveau encapsulé dans la structure de paquet du réseau suivant. Un datagramme va donc subir une série d'encapsulation/décapsulation jusqu'à ce qu'il atteigne l'ordinateur hôte destination;

- l'envoi et la réception de datagrammes d'une taille maximale fixée par le réseau qui correspond à la dimension maximale du champ de *données* du paquet de ce réseau;

- la transformation d'une adresse internet en une adresse sous-réseau;

- la réponse aux indications (erreur, contrôle de flux, ...) provenant du réseau.

Les cinq fonctions présentées ici plus haut correspondent aux fonctions de convergence du protocole SNDCP du modèle OSI.

Nous allons décrire ici plus bas les problèmes que pose l'interfaçage aux réseaux X.25, Ethernet et IEEE 802. Rappelons que le chapitre 6 est consacré à la description de l'implémentation des interfaces entre les niveaux accès-réseau, inter-réseaux et bout en bout pour la version 4.2BSD. L'interface au réseau qui y est décrit est l'interface Ethernet.

2.3.4.2.1. Interface au réseau X.25

Le problème le plus délicat lié à l'interfaçage du réseau X.25 est celui de l'harmonisation des services. Les réseaux X.25 offrent en effet un service avec connexion alors que le protocole IP utilise un service sans connexion.

La solution adoptée pour ce problème consiste à vérifier chaque fois qu'il arrive un datagramme du module IP, s'il existe déjà un circuit virtuel pour la destination du datagramme. Si oui, on utilisera ce circuit. Un nouveau circuit ne sera établi que s'il n'existe encore aucun circuit virtuel pour la destination du datagramme ou si la dimension des files d'attente pour les circuits existant vers cette destination dépasse une limite donnée. La libération d'un circuit virtuel s'effectuera à l'expiration d'un délai d'attente après l'expédition du dernier paquet.

Le datagramme à envoyer sera encapsulé dans un paquet de données X.25. Ce paquet sera ensuite transmis sur le circuit virtuel approprié. Arrivé à l'ETTD destination, le datagramme internet sera décapsulé du paquet de données X.25. La taille normale du champ de données des paquets échangés entre les deux ETTD a été fixée à 576 octets. A l'intérieur du réseau, ce paquet sera fragmenté en paquet X.25 de 128 octets. Cette fragmentation restera toutefois

invisible aux deux ETTD's. Ils pourront éventuellement utiliser les services complémentaires du réseau X.25 pour augmenter cette valeur de 576 octets.

Le problème de la mise en correspondance des adresses internet avec les adresses X.121 est généralement résolu par l'emploi d'un répertoire d'adresses.

Le traitement des erreurs et anomalies constatées suite à l'arrivée de paquets de réinitialisation, de reprise ou de diagnostic, doit également être pris en charge par la fonction d'harmonisation des services.

2.3.4.2.2. Interface aux réseaux Ethernet et IEEE 802

Dans le cas des réseaux Ethernet et IEEE 802, la fonction d'harmonisation des services est triviale puisque le service offert par ces réseaux correspond à ce que le protocole IP demande.

Pour de réseau EThernet, le datagramme internet sera simplement encapsulé dans un paquet Ethernet. La valeur du champ *type de paquet* (16 bits) sera 0 X 0800 pour identifier le protocole utilisateur, en fait IP. Arrivé à la station destination, le datagramme sera décapsulé et passé au module IP.

Pour les réseaux IEEE 802, on ajoutera au datagramme une en-tête SNACP(Sub-Network Access Protocol) qui définira le type de réseau utilisé et le module à qui adresser le datagramme, une en-tête LSAP (Link Service Access Point) et une en-tête MAC (Medium Access Control).

Pour les réseaux Ethernet et IEEE 802, la taille maximale de leur champ de *données* est fixée à 1500 octets.

Le problème de la conversion des adresses internet et Ethernet est résolu par l'emploi du protocole ARP décrit au paragraphe 2.3.2. Pour les réseaux IEEE, nous ne connaissons pas la solution adoptée.

Si la station destination a détecté une erreur de transmission (somme de contrôle inexacte), le paquet est purement et simplement détruit. Les problèmes provoqués par des paquets détruits, dupliqués ou reçus hors séquence, seront pris en charge par le protocole TCP.

Pour résumer, nous avons présenté à la figure 2.9. un exemple de mise en oeuvre du protocole IP entre les réseaux X.25 et Ethernet.

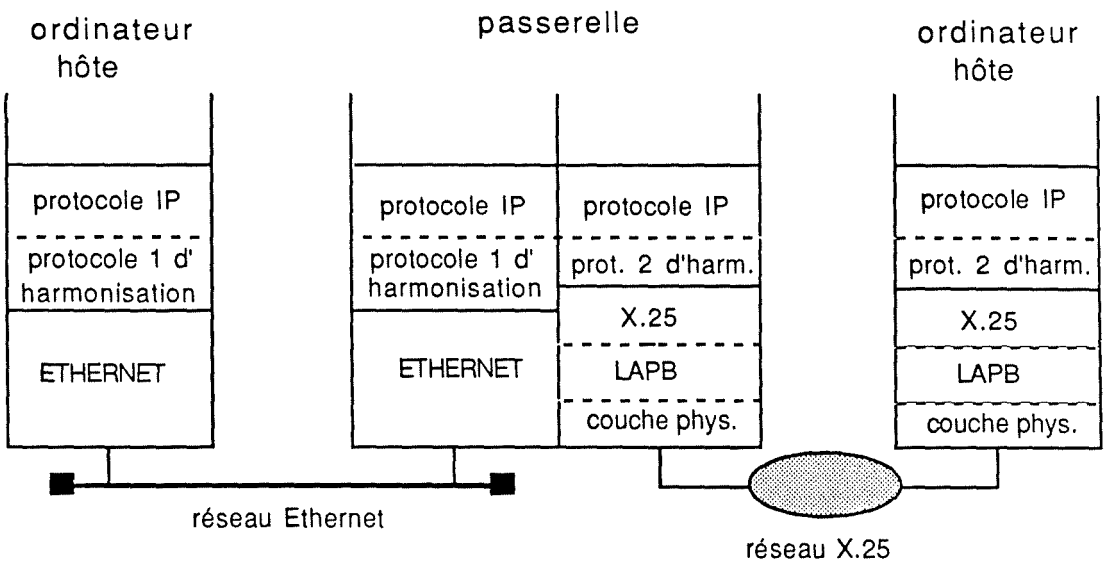


Fig. 2.9. - Exemple de mise en oeuvre du protocole IP.

Cet exemple illustre l'organisation générale des protocoles de niveaux accès-réseau et inter-réseaux pour l'architecture DoD. A un niveau accès-réseau, nous avons voulu faire apparaître les différences de services offerts par les réseaux par des hauteurs différentes pour les niveaux accès-réseau. Au niveau inter-réseaux, nous avons voulu distinguer les fonctions d'harmonisation spécifiques à chaque réseau des fonctions du protocole IP qui sont communes à tous les réseaux. Cette dernière distinction est artificielle puisque, en pratique, le protocole IP inclut également les fonctions d'harmonisation.

2.4.3.4. *Protocoles exigés dans les passerelles*

Dans ce point, nous allons donner la liste des protocoles qu'une passerelle doit implémenter; les protocoles qui ne font pas l'objet d'une présentation ailleurs dans ce travail, y seront décrits sommairement.

2.3.4.3.1. Les protocoles accès-réseau

Pour chaque réseau auquel la passerelle est connectée, il est évident que celle-ci doit implémenter le protocole pour y accèder.

2.3.4.3.2. Le protocole IP

Le protocole IP est l'objet de ce chapitre. Rappelons quelles sont les principales fonctions du protocole IP activées lors de l'arrivée d'un datagramme à une passerelle : il s'agit du routage de ce datagramme,en ce, compris la prise en charge des options de routage source et d'enregistrement du parcours, et

l'éventuelle fragmentation des datagrammes trop longs. Le réassemblage des datagrammes est généralement réalisé par l'ordinateur hôte destination.

2.3.4.3.3. Le protocole ICMP

Le protocole ICMP doit être implémenté chaque fois que le protocole IP est présent. Il permet de renseigner l'ordinateur hôte source sur les problèmes rencontrés lors de la transmission d'un datagramme ou de lui fournir des informations relatives au comportement du catenet. On distingue donc deux types de messages ICMP : le message ICMP de rapport d'erreur et le message ICMP d'information. Le premier prend en compte les évènements suivants : réseau ou ordinateur hôte hors de portée, temps dépassé, problème de paramètrage, congestion, message de redirection. Le second a trait aux informations suivantes : écho et réponse d'écho, estampille, demande d'information. La description de ce protocole est l'objet du chapitre suivant.

2.3.4.3.4. Les protocoles IGP

Pour comprendre l'intérêt des protocoles IGP (Interior Gateway Protocol), nous devons préalablement définir la notion de système autonome. Un **système autonome** est un sous-ensemble de passerelles du catenet qui ont été regroupées pour des raisons techniques, organisationnelles ou politiques. Les passerelles appartenant à un système autonome présentent les caractéristiques suivantes :

- la tutelle d'une organisation unique responsable du fonctionnement et de la maintenance du système autonome;

- l'emploi d'un protocole de routage dynamique distribué commun.

Puisqu'il existe différents protocoles de routage dynamique distribué, il faudra en choisir un pour le système autonome. Citons parmi ces protocoles, les protocoles RIP, GGP (Gateway-to-Gateway Protocol) et SPF (Shortest-Path-First). Protocole IGP est le terme générique donné pour désigner l'ensemble des protocoles de routage dynamique distribué disponibles pour gérer les tables de routage d'un système autonome.

Ces protocoles comprennent généralement des algorithmes

- pour définir la longueur de chaque chemin du catenet et
- pour calculer le chemin le plus court à une destination donnée.

Le protocole RIP nous est déjà familier. Pour mesurer la longueur des chemins, il utilise un attribut métrique qui indique le nombre de passerelles qui doivent être traversées par un datagramme pour aller à la destination du chemin. La valeur particulière 16 indique que le réseau est hors de portée. Pour mettre à jour ses propres tables de routage, certaines passerelles[5] testent régulièrement si chacune de ses voisines est encore accessible ou est devenue hors de portée. Pour mettre à jour les tables de routage des passerelles voisines, ces passerelles leur envoyent périodiquement ses propres tables.

---

5 Il s'agit des passerelles inter-réseaux.

Le protocole GGP utilise pour définir la longueur d'un chemin le nombre de passerelles séparant deux passerelles. Chaque modification apportée à la configuration du système autonome entraîne des changements dans toutes les tables de routage. Ils vont y être apportés de la manière suivante : la passerelle qui détecte la modification, va envoyer une mise à jour uniquement aux passerelles voisines. Cette mise à jour contiendra une entrée pour chaque réseau connu, indiquant le nouveau nombre minimum de passerelles entre ce réseau et la passerelle qui émet la mise à jour.

Dans le protocole SPF, chaque passerelle dispose d'une copie des tables de routage qui décrivent le système autonome. La longueur des chemins est mesurée de la même manière que dans le protocole GGP. Pour construire ces copies, la passerelle qui détecte un changement envoit ses mises à jour à toutes les passerelles du système autonome. Ces mises à jour reprennent uniquement la liste des (nouvelles) distances pour chacune des passerelles voisines.

Le protocole SPF présente par rapport au protocole GGP les deux avantages suivants :

- une reconfiguration plus rapide au système autonome puisque les mises à jour sont envoyées directement à toutes les passerelles du système autonome;

- un volume de mise à jour plus réduit puisque, dans le protocole SPF, on ne reprend que la liste des distances de la passerelle qui a constaté la modification à chacune de ses voisines.

2.3.4.3.5. Le protocole EGP

Comme un datagramme internet avant d'arriver à destination peut transiter par des passerelles appartenant à des systèmes autonomes différents, et comme les systèmes autonomes doivent également s'échanger des informations de routage, on va utiliser un protocole EGP (Exterior Gateway Protocol) qui organisera l'échange d'informations de routage entre passerelles des différents systèmes autonomes.

L'objectif du protocole EGP est de garantir une bonne utilisation des ressources du catenet en s'assurant, par exemple, qu'un réseau particulier déclaré hors de portée à l'intérieur de son système autonome par le protocole IGP, le soit également dans les autres systèmes autonomes.

Le premier protocole EGP défini dans (RFC 904), était très dépendant de la topologie des systèmes autonomes américains, qui sont pratiquement tous connectés aux réseaux Arpanet ou Milnet. Un nouveau protocole EGP a été défini par (RFC 975) en 1986. Ce protocole cherche à réduire les restrictions d'ordre topologique imposées par le précédent. Ce type de protocole fait l'objet actuellement de nombreuses recherches outre-Atlantique.

2.3.4.3.6. Le protocole IGMP

Le protocole IGMP (Internet Group Management Protocol) est une extension du protocole IP définit pour fournir l'adressage en mode diffusion au niveau du catenet. Il permet donc de transmettre des copies d'un même datagramme internet à un ensemble d'ordinateurs hôtes du catenet. Ce protocole n'existe qu'à titre expérimental.

2.3.4.3.7. Les protocoles de fonctionnement et de maintenance

Le problème est le suivant. Une passerelle comme tout équipement informatique, peut en cours de fonctionnement rencontrer des problèmes hardware au niveau de son processeur, de ses interfaces, des équipements connectés, etc. Son logiciel comme ses équipements peuvent aussi être modifiés. Elle devra, en outre, contrôler son fonctionnement et ses performances de manière à éviter éventuellement des situations de congestion. Bref, tout au long de la vie de la passerelle, une série d'événements et d'incidents peut survenir.

Pour réagir à ces incidents, il existe toute sorte de solutions. A une extrémité, la solution apportée peut être exclusivement locale. On peut, par exemple, disposer d'une personne qui reste sur place pour prendre en charge les problèmes qui se posent. A l'autre extrémité, la solution peut prévoir que la plupart des contrôles et des interventions soient faites à distance. La première solution est adéquate quand le nombre de passerelles à superviser est réduit. Dans le cas de configurations importantes, la seconde solution est plus appropriée.

Pour réaliser ces opérations à distance, on utilise des protocoles de fonctionnement et de maintenance. Ceux-ci utilisent généralement les services de transport offerts par les protocoles TCP ou UDP, ce qui oblige donc les passerelles à implémenter également ces protocoles.

Les protocoles de fonctionnement et de maintenance font encore actuellement l'objet de nombreuses recherches. A ce jour, aucun d'entre eux n'a encore atteint le stade de standard internet.

En résumé, parmi tous les protocoles qui viennent d'être énoncés, on doit distinguer les catégories suivantes :

- les protocoles qui doivent obligatoirement être présents dans chaque passerelle du catenet; il s'agit des protocoles IP et ICMP dont l'implémentation doit être conforme aux standards internet;

- les protocoles qui, dans un avenir proche, devront être présents dans chaque passerelle du catenet; il s'agit des protocoles EGP et IGMP pour lesquels on attend la définition d'un standard internet;

- les protocoles qui doivent être implémentés dans chaque passerelle du système autonome; il s'agit des protocoles IGP et des protocoles de fonctionnement et de maintenance. Leur choix relève d'une autorité locale.

Rappelons que certains de ces protocoles utilisent les services de transport offerts par les protocoles UDP ou TCP. Ceci entraîne que ces protocoles de niveau bout en bout doivent également être implémentés dans chaque passerelle.

2.3.4.4. *Les types de passerelles*

Lors de la présentation du protocole RIP, nous avons déjà eu l'occasion de distinguer deux types de passerelles en fonction de leur rôle dans ce protocole. Il s'agissait des passerelles normales et des passerelles inter-réseaux. Auparavant, nous avions signalé que la version 4.2BSD offrait la possibilité à un ordinateur hôte de devenir également une passerelle. On peut donc ainsi différentier les passerelles spécifiques des passerelles ordinateurs hôtes : les premières n'implémentent que les protocoles exigés, c'est-à-dire ceux qui ont été décrits au point précédent, tandis que les secondes y ajoutent tous les protocoles de niveau applications.

Dans la suite, nous allons présenter une troisième classification dans laquelle on répartit les passerelles suivant leur manière de réaliser l'interconnexion de réseaux tout en restant dans le cadre défini par le DoD. On distingue ainsi les ponts, les passerelles et les convertisseurs de protocoles.

2.3.4.4.1. Les ponts

Le **pont** est une passerelle simplifiée utilisée pour réaliser l'interconnexion de réseaux homogènes. Deux réseaux sont homogènes s'ils ont des méthodes d'accès (sous-couche MAC) éventuellement différentes mais des protocoles de niveau supérieur (sous-couche LLC) identiques. Le pont est particulièrement bien adapté pour interconnecter des réseaux locaux.

Un pont qui connecte deux réseaux A et B opère de la manière suivante:

- il lit toutes les trames transmises sur A et il sélectionne celles adressées au réseau B;
- il transmet celles-ci sur le réseau B;
- il fait la même chose pour le trafic entre B et A.

La figure 2.10 représente un exemple d'interconnexion d'un réseau Ethernet à un réseau Token Ring.

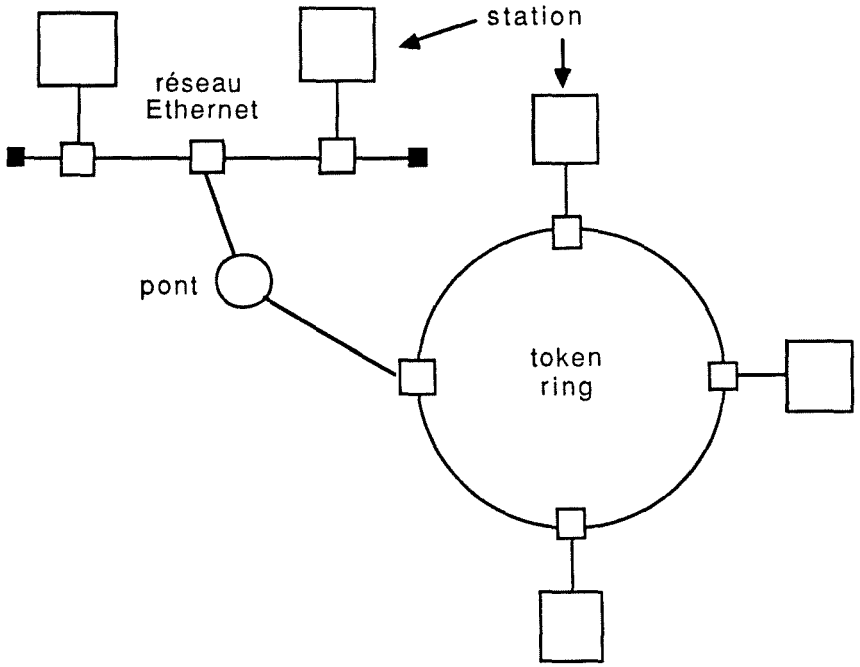


Fig. 2.10. - Exemple d'interconnexion de réseaux locaux.

Ces deux réseaux ont des méthodes d'accès et des formats de trame différents. Pour les interconnecter, le pont ne va pas utiliser les mécanismes de routage et d'encapsulation du protocole IP. Au lieu de cela, il va implémenter des méthodes de routage ainsi que de conversion de trames qui dépendront toujours des réseaux connectés.

Ainsi, le pont devra disposer de fonctions d'adressage et de routage pour pouvoir déduire de l'adresse destination contenue dans l'en-tête de la trame, à quel réseau la trame est adressée. Remarquons bien que le pont prend ses décisions de routage sur base de l'adresse contenue dans l'en-tête de la trame alors que la passerelle utilise l'adresse internet contenue dans l'en-tête du datagramme.

Quand un pont relie deux réseaux dont les formats de trame diffèrent, il doit pouvoir également adapter le format des trames, éventuellement transformer l'ordre d'envoi des bits et recalculer la zone de détection d'erreur. De ceci, il apparaît clairement que le pont n'utilise pas la technique de l'encapsulation.

2.3.4.4.2. Les passerelles

En introduction, nous avions déjà signalé que l'objectif poursuivi par le Département de la Défense était d'interconnecter des réseaux hétérogènes. Pour y parvenir, la solution retenue était celle de la passerelle. Cette solution présente l'avantage de tolérer aussi la présence de ponts dans le catenet à la condition que les réseaux locaux interconnectés par le(s) pont(s) aient un numéro de réseau commun.

Une passerelle est un équipement physique qui constitue un point unique de transit de données entre deux ou plusieurs réseaux. Elle assure essentiellement les fonctions de routage et d'adressage. Elle peut aussi être nécessaire pour des besoins de gestion quand on veut contrôler le trafic entre deux réseaux.

2.3.4.4.3. Les convertisseurs de protocoles

Le modèle DoD n'est malheureusement pas le seul système qui permet le partage des ressources. Des constructeurs (IBM, DEC, etc.) ainsi que des organisations internationales (ISO, CCITT, etc.) proposent aussi des systèmes (SNA, DNA, OSI, etc.) qui rendent possible ce partage. Le nouveau problème qui se pose est le suivant : comment est-il possible à partir d'un système donné, de disposer des ressources gérées par un autre système ?

Pour lever cet obstacle, on a tendance à utiliser des convertisseurs de protocoles. "*Cette technique consiste à utiliser des systèmes (au sens ordinateur) qui implémentent tout ou partie de deux architectures de protocoles jusqu'à un niveau à un mécanisme de correspondance assure la traduction (unilatérale ou bilatérale) d'un protocole dans un autre*" (Fluckiger 87, p. 15-5). "*Un convertisseur comprend donc (au moins) deux architectures distinctes jusqu'au niveau où est effectuée la conversion*" (Fluckiger 87, p. 15-7). Actuellement, par exemple, un programme de convertisseurs de protocoles a été développé au CERN pour effectuer la conversion entre les protocoles de transfert de fichier des architectures CERNET, DECNET, JANET, TCP-IP, ISO (FTAM). D'autres convertisseurs sont opérationnels autour du protocole de messagerie X.400.

2.4. Mécanismes de fragmentation et de réassemblage des datagrammes

2.4.1. Présentation du problème

En cours d'acheminement, un datagramme peut être amené à traverser des réseaux dont la taille maximum du champ de données des paquets est inférieure à la taille du datagramme. Pour surmonter ce problème, le protocole IP fournit un mécanisme de fragmentation qui peut être utilisé par l'ordinateur hôte source d'un datagramme mais aussi par les passerelles visitées. L'éventuel réassemblage du datagramme ne s'effectuera qu'à l'ordinateur hôte destination.

Dans la suite de l'exposé, on utilisera le terme datagramme initial pour désigner le datagramme qui renferme la totalité des données que l'utilisateur du service veut transmettre. Un datagramme fragmenté est un datagramme qui ne transporte qu'une partie des données du datagramme initial et dont l'en-tête est pratiquement identique à celle du datagramme initial.

Le reste de cette section va être organisé de la manière suivante :

- au paragraphe 2.4.2., nous allons décrire les champs de l'en-tête du datagramme utilisés par les fonctions de fragmentation et de réassemblage;

- au paragraphe 2.4.3., nous présenterons les traitements qu'entraînent les opérations de fragmentation et de réassemblage;

- au paragraphe 2.4.4., on comparera les choix effectués pour cette fonction dans le protocole IP à ceux faits dans d'autres modèles. C'est aussi dans ce paragraphe que nous présenterons la solution proposée par ISO dans le protocole ISO 8473 en matière de segmentation et de réassemblage.

2.4.2. Champs de l'en-tête du datagramme utilisés pour les fonctions de fragmentation et de réassemblage

On répartit généralement ces champs suivant le rôle qu'ils jouent dans les fonctions de fragmentation et de réassemblage.

Pour s'assurer que les différents datagrammes fragmentés ne soient pas mélangés à la phase de réassemblage, on va identifier le datagramme initial avec les quatre champs suivants :

- les *adresses internet source et destination* (chacune 32 bits);

- l'*identificateur de protocole* (8 bits) pour désigner le protocole qui a demandé d'envoyer les données contenues dans le champ de données du datagramme;

- l'*identificateur de datagramme* (16 bits) dont la valeur doit rester unique pour la période pendant laquelle le datagramme va séjourner sur le catenet.

Le choix d'associer l'identificateur de protocole et l'identificateur de datagramme ne se justifie que si le protocole utilisateur d'IP peut choisir l'identificateur de datagramme. Si, par exemple, pour obtenir une communication fiable, ce protocole supérieur travaille par réémission des données, on va augmenter la probabilité d'une réception correcte s'il utilise le même identificateur de datagramme. Ainsi, le protocole IP pourra utiliser les datagrammes provenant de deux émissions pour reconstituer le datagramme initial correct.

A côté de ces quatre champs qui permettent d'identifier le datagramme initial, il existe quatre autres champs qui rendent possible le réassemblage du datagramme initial par l'ordinateur hôte destination. Ces champs sont les suivants :

- le *drapeau MF* (More Fragment, 1 bit) qui indique que le datagramme transporte (MF = 0) ou ne transporte pas (MF = 1) le dernier octet du champ de données de datagramme initial;

- le *décalage* (= offset, 13 bits) qui indique, en multiple de 64 bits, la position des données du datagramme dans le datagramme initial; une valeur zéro indique que le datagramme transporte le premier octet du champ de données du datagramme initial;

- la *longueur totale du datagramme* (16 bits) qui mesure en octets la longueur de l'en-tête et du champ de données du datagramme. Ce champ autorise le datagramme initial à avoir une longueur totale de 65 565 octets ($2^{16} - 1$). La plupart des ordinateurs hôtes et la plupart des réseaux ne tolèrent toutefois pas des datagrammes d'une telle longueur. C'est pourquoi, en pratique, le protocole IP impose que chaque ordinateur hôte soit capable d'accepter des datagrammes d'une longueur de 576 octets. Ceux-ci peuvent arriver entiers ou fragmentés. Si un ordinateur hôte veut quand même envoyer un datagramme plus long, il doit être sûr que l'ordinateur hôte destination puisse l'accepter. La longueur de 576 octets a été choisie parce qu'elle réserve suffisamment d'espace pour l'en-tête du protocole IP (généralement 20 octets), l'en-tête du protocole de bout en bout (généralement 20 octets pour TCP) et l'en-tête et les données du protocole application;

- la *longueur de l'en-tête du datagramme* (4 bits) qui mesure, en multiple de 4 octets, la longueur de l'en-tête (options comprises) du datagramme. Ce champ permet d'indiquer où commence le champ de données du datagramme. Sa valeur minimum est 5 (20 octets), sa valeur maximum est 15 (60 octets).

Il est également possible d'empêcher de fragmenter un datagramme. Cela est réalisé au moyen du *drapeau DF* (Don't Fragment, 1 bit). Si, au cours de son périple vers l'ordinateur hôte destination, un datagramme avec le drapeau DF égal à 1 doit être fragmenté, il sera purement et simplement détruit.

2.4.3. Opérations de fragmentation et de réassemblage

2.4.3.1. *La fragmentation d'un datagramme*

La fragmentation d'un datagramme internet est nécessaire quand le datagramme doit traverser un réseau qui limite la taille de ses paquets à une dimension inférieure à celle du datagramme reçu par la passerelle ou du datagramme à envoyer par l'ordinateur hôte.

Si le drapeau DF du datagramme est mis à un, ce datagramme sera détruit. Cette option a été introduite dans le protocole IP pour empêcher de fragmenter un datagramme dans le cas où l'ordinateur hôte source sait que l'ordinateur hôte destination n'a pas assez de ressource pour reconstituer le datagramme initial.

Si le drapeau DF est mis à zéro, on pourra créér le nombre nécessaire de nouveaux datagrammes. Supposons ce nombre égal à n. Ensuite, on va recopier l'en-tête du datagramme reçu dans l'en-tête de chaque nouveau datagramme. Les options du datagramme reçu seront recopiées dans l'en-tête du premier datagramme fragmenté in extenso. Pour les autres datagrammes, certaines options ne seront pas recopiées. Le paragraphe 2.5.5. explique comment les options sont traitées en cas de fragmentation. Les données du datagramme reçu seront fragmentées en n parties. Les n-1 premières parties possèderont une longueur multiple de huit octets. La dernière sera la seule à ne pas avoir nécessairement une longueur multiple de huit octets. Ces n parties de données seront alors replacées dans leur datagramme respectif.

Pour chaque nouveau datagramme, on va recalculer la longueur totale et la longueur de l'en-tête. Le drapeau MF sera mis à un pour tous les datagrammes fragmentés excepté le dernier. Le champ décalage sera recalculé sauf pour le premier datagramme dont la valeur restera zéro.

Pour fragmenter les données du datagramme reçu en n parties, chaque module IP est libre de le faire comme il l'entend. Dans la version 4.2BSD, la méthode choisie consiste à utiliser pour les n-1 premières parties la dimension maximale autorisée par le réseau. Une autre méthode serait de diviser les données du datagramme reçu en deux un certain nombre de fois de manière à obtenir de nouveaux datagrammes qui peuvent être encapsulés dans le champ de données des paquets.

Remarquons encore que chaque réseau du catenet doit être capable de transmettre des datagrammes de 68 octets sans avoir à les fragmenter. Cette longueur est obtenue en additionnant la longueur maximum de l'en-tête d'un datagramme (60 octets) et la longueur minimum de son champ de données (8 octets).

2.4.3.2. *Le réassemblage du datagramme initial*

Un datagramme initial n'est reconstitué qu'une fois arrivé à l'ordinateur hôte destination. Il n'est donc pas réassemblé à chaque passerelle intermédiaire. De cette manière, le travail des passerelles est grandement simplifié et on évite les ralentissements que ces traitements entraîneraient. De plus, les différents datagrammes générés peuvent suivre des itinéraires différents.

Pour reconstituer le datagramme initial, le module IP regroupe les datagrammes reçus suivant les quatre champs d'identification. Leurs données sont insérées en bonne position dans un espace de stockage réservé pour le datagramme initial.

Si le datagramme reçu a un drapeau MF égal à zéro et un décalage nul, il s'agit d'un datagramme non fragmenté. Le traitement de réassemblage s'arrête déjà ici. Sinon, le datagramme initial ne sera entièrement réassemblé que quand on aura reformé l'ensemble contigu des données et reçu le datagramme dont le drapeau MF est égal à zéro.

Pour éviter que des datagrammes incomplets n'occupent indéfiniment de l'espace de stockage, on a introduit le concept de **temps limite de réassemblage**. Dans la version 4.2BSD, ce concept a été implémenté de la manière suivante : quand un nouveau datagramme arrive à destination, on lui associe un temps limite de réassemblage. Le plus souvent, ce délai est fixé à 15 secondes. Si, passé ce délai, le datagramme initial n'a pas été reconstitué, son espace de stockage sera libéré. On pourrait croire que ce délai a été fixé de manière autonome. Ce n'est toutefois pas le cas. Ce délai est en effet étroitement lié à la capacité de stockage de l'ordinateur hôte destination ainsi qu'à la vitesse de transmission du réseau.

2.4.4. Autres solutions possibles

2.4.4.1. *Introduction*

La question de savoir où doit se réaliser le réassemblage des datagrammes a surtout été débattue fin des années septante (Shoch 79). Deux tendances sont apparues. La première préconise que le datagramme initial ne doit être reconstitué qu'une fois arrivé à destination. Ce type de solution s'appelle **fragmentation inter-réseaux**. C'est cette solution qui a été adoptée par le modèle DoD. La seconde tendance propose que le datagramme initial soit reconstruit à la sortie de chaque réseau. Cette solution s'appelle **fragmentation intra-réseau**. Elle a été choisie par le modèle XNS de Xerox (Shoch 79, pp. 3-9).

Le reste de ce paragraphe va se diviser en deux points : le point 2.4.4.2. va dresser la liste des avantages et inconvénients de la fragmentation intra-réseau et de la fragmentation inter-réseaux; le point 2.4.4.3. présentera brièvement la solution que l'ISO propose dans son protocole ISO 8473 en matière de segmentation et de réassemblage. La segmentation est la terminologie ISO pour désigner le mécanisme de la fragmentation.

2.4.4.2. Avantages et inconvénients des fragmentations intra-réseau et inter-réseaux

La fragmentation intra-réseau présente par rapport à la fragmentation inter-réseaux les avantages suivants (Shoch 79, p. 4) :

- la manière de résoudre le problème de fragmentation et du réassemblage peut être spécifique à chaque réseau et donc être intégrée au niveau accès-réseau;

- il n'est pas nécessaire de retransmettre dans chaque paquet l'en-tête du datagramme internet;

- si, sur le parcours du datagramme, on utilise la fonction de fragmentation pour un seul réseau, toutes les parties de ce datagramme seront recollées à la sortie de ce réseau. Cela permet d'éviter un effet de cascade.

Cette solution présente les inconvénients suivants (Shoch 79, p. 4) :

- le traitement à effectuer par les passerelles s'est sensiblement complexifié puisque, outre les fonctions de routage et de fragmentation, il faut y ajouter celle de réassemblage;

- il n'est pas possible de tirer parti de l'existence de passerelles alternatives et poursuivre ainsi des itinéraires différents puisque tous les paquets doivent aboutir à la même passerelle de sortie pour y reconstituer le datagramme initial;

- si le datagramme doit traverser un grand nombre de réseaux et si, pour chacun de ces réseaux, le datagramme doit être fragmenté et réassemblé, ceci entraîne de nombreuses opérations de fragmentation et de réassemblage inutiles.

La solution d'une fragmentation inter-réseaux met en avant les atouts suivants (Shoch 79, p. 5) :

- la passerelle ne doit remplir que des fonctions de routage et de fragmentation;

- chaque datagramme peut suivre, à partir du moment où il est créé, un itinéraire différent de ceux suivis par les autres datagrammes créés au même moment.

Cette solution présente également des inconvénients. Ceux-ci sont les suivants :

- toutes les passerelles du catenet doivent suivre les mêmes règles de fragmentation;

- l'en-tête du datagramme initial doit être reproduit en grande partie dans chaque datagramme fragmenté.

Cette comparaison ne permet pas de rejeter un mécanisme en faveur d'un autre; c'est sans doute la raison pour laquelle il existe de nombreuses situations où les deux mécanismes se côtoyent. Nous allons ici en présenter deux.

Dans la première situtation, nous allons remarquer que la fragmentation intra-réseau peut être utilisée dans le modèle DoD tant qu'elle reste invisible au module IP. Cette situation peut se présenter lorsqu'un datagramme internet doit transiter par un réseau Arpanet. Arpanet offre en effet un service de transport de messages d'une longueur maximum de 8192 bits entre les ordinateurs qui lui sont connectés. Le transport de ces messages à l'intérieur d'Arpanet est réalisé en les fragmentant en paquets de 0 à 1024 bits. Ces paquets sont acheminés indépendamment les uns des autres et réassemblés dans le commutateur (appelé IMP, Interface Message Processor) d'arrivée, avant livraison à l'ordinateur destinataire (Macchi 87, p. 337). Le mécanisme de fragmentation est une fonction du protocole IMP-IMP (inter-commutateurs) d'Arpanet. Il reste tout à fait invisible aux modules IP des ordinateurs hôtes qui utilisent le service de transport d'Arpanet.

Dans la seconde situation, nous allons voir que le module IP peut utiliser la méthode de fragmentation intra-réseau disponible dans l'architecture XNS. Cette architecture présente comme le modèle DoD la caractéristique d'avoir au niveau inter-réseaux, un protocole unique, le protocole PUP. Le protocole PUP est donc l'équivalent du protocole IP dans l'architecture XNS. Pour traverser un réseau particulier, le datagramme PUP comme le datagramme IP utilisent la technique de l'encapsulation.

Ces deux propriétés communes ont permis de définir une nouvelle technique d'interconnexion de systèmes, à savoir l'**encapsulation mutuelle** (Shoch 81). Cette technique est applicable dans le cas où deux réseaux, d'architecture XNS, par exemple, sont interconnectés par un troisième réseau, d'architecture DoD. Dans ce cas, l'équipement d'interconnexion, parfois appelé relais, va encapsuler dans un datagramme internet chaque datagramme PUP reçu d'un des réseaux XNS connectés et transmettre le tout à travers le réseau intermédiaire DoD vers le relais de sortie. Celui-ci décapsulera le datagramme internet pour en extraire le datagramme PUP. Si nous généralisons cet exemple, la technique de l'encapsulation mutuelle consiste à considérer le catenet du modèle DoD comme un réseau particulier du système d'interconnexion de Xerox et vice versa.

Une conséquence de ce mécanisme est d'offrir au module IP présent sur un de ces relais, la possibilité d'utiliser la méthode de fragmentation intra-réseau disponible avec le protocole PUP. Remarquons que cela nécessite une gestion des tables de routage commune aux deux architectures.

2.4.4.3. Segmentation et réassemblage dans le protocole ISO 8473

Les propositions contenues dans le projet ISO/DIS 8473 présentent de nombreuses analogies avec le protocole IP en matière de segmentation et de réassemblage. Ceci explique pourquoi nous ne présenterons ces mécanismes que de manière très sommaire. Le tableau 2.2. résume les principaux résultats de la comparaison de ces deux protocoles.

Avant de présenter les principaux résultats de cette comparaison, rappelons qu'une NPDU correspond au concept de datagramme dans le protocole IP. Nous n'avons pas encore jusqu'ici définir le concept de segment. Un **segment** est une partie des données du champ de *données* de la NPDU initiale.

Pour identifier tous les segments correspondant à une NPDU initiale, on utilise les champs d'identificateur de la NPDU, d'adresse source et d'adresse destination. Contrairement au protocole IP, on ne laisse pas ici la possibilité à l'utilisateur de service de choisir l'identificateur de la NPDU.

Au niveau du réassemblage de la NPDU initiale, on constatera que le *champ longueur de la NPDU* et le *drapeau de segmentation supplémentaire* correspondent exactement au champ longueur totale du datagramme et au drapeau MF. Le champ *décalage de segment* mesure en octets le décalage tandis que dans le protocole IP, le décalage se mesure en multiple de 8 octets. Enfin, le *champ longueur totale de la NPDU initiale* mesure en octets la longueur de l'en-tête et du champ des données de la NPDU initiale. Cette information a été introduite pour permettre à l'entité réseau destination d'allouer l'espace nécessaire au réassemblage de la NPDU initiale dès la réception de la première NPDU.

En ce qui concerne l'option de *segmentation autorisée*, elle est tout à fait analogue à l'option Don't Fragment du protocole IP.

En conclusion, les deux protocoles utilisent le même mécanisme de fragmentation inter-réseaux et pratiquement les mêmes informations pour le réassemblage.

PROTOCOLE IP		PROTOCOLE ISO 8473	
technique utilisée :	fragmentation inter-réseaux	technique utilisée :	fragmentation inter-réseaux
champs d'identification des datagrammes		champs d'identification des NPDU	
libellé des champs	format	libellé des champs	format
identificateur du datagramme	2 octets	identificateur de la NPDU	2 octets
adresse internet source	4 octets	adresse source	longueur variable
adresse internet destination	4 octets	adresse destination	longueur variable
identificateur de protocole	1 octet		
champs utilisés au réassemblage		champs utilisés au réassemblage	
libellé des champs	format	libellé des champs	format
longueur totale du datagramme	2 octets	longueur de la NPDU	2 octets
décalage	13 bits	décalage de segment	2 octets
drapeau MF	1 bit	drapeau de segment supplémentaire	1 bit
longueur de l' en-tête du datagramme	4 bits	longueur totale de la NPDU initiale	2 octets
champ facultatif		champ facultatif	
libellé du champ	format	libellé du champ	format
drapeau DF	1 bit	drapeau de segmentation autorisée	1 bit

Tab. 2.2. - Comparaison des mécanismes de fragmentation et de réassemblage des protocoles IP et ISO 8473.

2.5. Autres fonctions du protocole IP

2.5.1. Introduction

A côté des fonctions déjà étudiées dans les deux sections qui précèdent, le protocole IP implémente cinq autres fonctions; ces fonctions sont :

- le maintien de la qualité de service,
- le contrôle de la durée de vie des datagrammes,
- la détection d'erreur à l'en-tête du datagramme,
- l'utilisation d'options et
- le transport des données.

Ces cinq fonctions se retrouvent également dans le protocole ISO 8473. C'est pourquoi, dans les cinq paragraphes qui suivent, nous présenterons ces fonctions pour chacun des deux protocoles.

2.5.2. Le type de service

2.5.2.1. *Protocole IP*

Le champ *type de service* (TOS, pour Type Of Service, 8 bits) permet à l'utilisateur du service datagramme d'indiquer au fournisseur de service la qualité de service réseau qu'il souhaite. Cette information est générique dans le sens où elle est indépendante d'un réseau particulier. Quand, suite à une décision de routage, les ordinateurs hôtes ou les passerelles doivent choisir les paramètres actuels de qualité de service pour un réseau particulier, c'est sur les indications contenues dans le champ type de service que ce choix devra s'aligner.

Ce champ permet de nuancer la qualité du service demandé dans les quatre domaines suivants :

- la priorité, pour indiquer l'importance du datagramme; cette indication peut être utile si, sous l'effet d'une charge excessive, le réseau particulier n'accepte encore de transmettre que des paquets d'un certain niveau de priorité;

- le délai, pour indiquer la promptitude avec laquelle le service devra être rendu,

- le débit, pour indiquer la cadence de transfert que le réseau doit adopter;

- la fiabilité, pour indiquer le degré d'effort requis pour garantir la réussite de la transmission du datagramme.

Les huit bits qui composent ce champ ont la signification suivante :

bits 0-2 : la priorité (de 000 pour la priorité la plus faible à 111 pour une grande priorité);
bit 3 : 0 : délai normal,
1 : délai court;
bit 4 : 0 : débit normal,
1 : débit élevé;
bit 5 : 0 : fiabilité normale,
1 : grande fiabilité;
bits 6-7 : disponibles pour une utilisation future (mis à 00).

L'analyse de ces champs entraîne les observations suivantes :

- pour la plupart des réseaux, si de bons résultats sont obtenus pour un de ces paramètres, cela impliquera de moins bonnes performances pour un autre;

- puisque rares sont les réseaux qui permettent de satisfaire simultanément à toutes ces exigences, la qualité finale du service rendu ne sera déterminée que suivant les possibilités du réseau;

- si un réseau ne peut maintenir la qualité du service souhaitée, cela n'entraînera pas nécessairement la destruction du datagramme; ce choix relève en effet d'une décision locale;

- en pratique, sauf cas exceptionnels, seuls deux des bits 3, 4 et 5 peuvent être mis à 1.

2.5.2.2. *Protocole ISO 8473*

Dans le projet ISO/DIS 8473, le *maintien de la qualité de service* n'est disponible qu'en option. Les exigences que l'utilisateur du service peut exprimer portent sur le temps de transit, le coût et la probabilité d'erreur résiduelle. L'utilisateur du service va pouvoir exprimer ses préférences de la manière suivante :

- si les considérations de temps doivent l'emporter sur les considérations de coût, le 8e bit du champ *valeur de paramètre* (8 bits) pour l'option maintien de la qualité de service prend la valeur 1; une valeur zéro signifie que les considérations de coût l'emportent;

- si les décisions de routage doivent viser la probabilité d'erreur résiduelle la plus faible et non le temps de transit le plus bas, le 7e bit pour le champ *valeur de paramètre* prend la valeur 1; une valeur de zéro indique le contraire;

- si les décisions de routage doivent viser la probabilité d'erreur résiduelle la plus faible et non le coût le plus faible, le 6e bit prend la valeur 1; une valeur de zéro correspond au phénomène inverse;

- les bits 5 à 1 ne sont pas spécifiés (ISO/DIS 8473, p. 29).

2.5.3. La durée de vie d'un datagramme

2.5.3.1. *Protocole IP*

Le champ *durée de vie* d'un datagramme (TTL, pour Time To Live, 8 bits) indique le temps restant pendant lequel un datagramme peut encore séjourner dans le catenet. Ce champ est initialisé par l'émetteur du datagramme. La durée de vie est exprimée en secondes. A chaque module IP qui traite le datagramme, ce champ est décrémenté d'une unité, quel que soit ses temps de transfert et de traitement. Si sa valeur devient nulle avant que le datagramme n'ait rejoint sa destination, il sera détruit. En cas de fragmentation, le champ durée de vie est reproduit dans chaque datagramme fragmenté.

La raison d'être de la fonction durée de vie est double. Elle empêche, d'une part, qu'un datagramme ne circule indéfiniment dans le catenet et, d'autre part, elle offre à l'utilisateur du service datagramme un mécanisme qui permet de limiter la durée de vie d'un datagramme.

2.5.3.2. *Protocole ISO 8473*

Dans le protocole ISO 8473, il existe également une fonction *durée de vie* d'une NPDU qui diffère sur les deux points suivants de la fonction du protocole IP :

- la durée de vie d'une NPDU est exprimée en unités de 500 millisecondes;

- "*lorsqu'une entité de réseau traite une NPDU, elle décrémente la durée de vie d'au moins une unité. La réduction sera supérieure si la somme du temps de transit dans le sous-réseau d'où la NPDU provient et du temps propre au système traitant la NPDU dépasse ou est supposée dépasser 500 millisecondes. Dans ce cas, la durée de vie doit être décrémentée à raison d'une unité pour 500 millisecondes supplémentaires de délai*" (ISO/DIS 8473, p. 22). Le projet précise également qu' "*il n'est pas nécessaire pour chaque système intermédiaire de soustraire un mesure précise du temps écoulé depuis le départ de la NPDU du système intermédiaire précédent. Il suffit de soustraire une limite maximale du temps pris. Dans la plupart des cas, il suffit au système intermédiaire de soustraire une valeur constante, dépendant des délais-type, proches du maximum rencontré dans le sous-réseau donné. Une estimation précise ne s'impose que pour les sous-réseaux caractérisés à la fois par un temps maximal relativement élevé et des variations importantes de délais*" (ISO/DIS 8473, p. 53).

2.5.4. La somme de contrôle de l'en-tête

2.5.4.1. *Protocole IP*

Le champ *somme de contrôle* de l'en-tête (16 bits) permet de détecter les erreurs de transmission influançant l'en-tête du datagramme mais non son champ de données. Cette somme de contrôle est vérifiée à chaque module IP. Puisque certains champs tels que la durée de vie ou ceux utilisés par la fonction de fragmentation changent à chaque passerelle, elle y est également recalculée. Si la vérification de la somme de contrôle signale une erreur, le datagramme est détruit par le module IP qui l'a détectée.

2.5.4.2. *Protocole ISO 8473*

Dans le protocole ISO 8473, la fonction de *détection d'erreur* est identique à celle du protocole IP. On notera toutefois que l'on peut indiquer si l'on veut tenir compte ou non du résultat du contrôle. "*Une valeur zéro du champ somme de contrôle de l'en-tête de la NPDU sert à indiquer qu'on ne doit pas tenir compte de la somme de contrôle. La fonction de détection d'erreur à l'en-tête permet de s'assurer que la valeur zéro ne représente pas une somme de contrôle valide. Une valeur autre que zéro indique que l'on doit traiter la somme de contrôle*" (ISO/DIS 8473, p. 23).

2.5.5. Les options

2.5.5.1. *Protocole IP*

Par options, on entend des champs qui peuvent apparaître dans l'en-tête de datagramme suite à une demande formulée par l'utilisateur de service ou suite à une décision locale. Les fonctions associées à ces options doivent être implémentées dans chaque module IP. Ce qui est optionnel, c'est en fait leur transmission dans un datagramme particulier mais pas leur implémentation[6] (RFC 791, p. 15).

La longueur du champ des options est naturellement variable. Zéro ou plusieurs options sont seulement possibles. Une option commence toujours au début d'un mot de quatre octets. Chaque option adopte l'un des deux formats suivants :

- un octet unique de type d'option;
- un octet de type d'option, un octet de longueur d'option et les octets de données de l'option. La longueur d'option mesure en octets la longueur du type, de la longueur et des données de l'option.

---

[6] Signalons que c'est loin d'être le cas dans la version 4.2BSD du modèle DOD. Ainsi, les options *sécurité* et *enregistrement de la route* ne sont pas du tout prises en charge par le module IP.

L'octet de type de l'option est composé des trois champs suivants :

- bit 0 : le drapeau de copie;
- bits 1-2 : la classe d'option;
- bits 3-7 : le numéro d'option.

Le drapeau de copie mis à 1 indique que cette option doit être recopiée dans tous les segments lors de la fragmentation. Mis à zéro, l'option ne sera copiée que dans le premier datagramme. La classe d'option est mise à 00 quand il s'agit d'options de contrôle. Elle prend la valeur 10 pour les options de mesure et de débuggage. Toutes les options définies actuellement dans le protocole IP appartiennent à la classe 00 à l'exception de l'option d'estampillage qui relève de la classe 10. Les classes d'options 01 et 11 sont réservées à un emploi futur (RFC 791, p. 15).

C'est en passant en revue les huit options disponibles avec le protocole IP que nous définirons les valeurs prises par le champ numéro d'option puisqu'il permet de les identifier.

2.5.5.2. *La fin de la liste des options*

Format : octet de type d'option = 00000000

Cet octet indique la fin de la liste des options. Cette option ne peut être utilisée que pour faire correspondre la fin de la liste des options avec la fin d'un mot de quatre octets. En pratique, il n'y a aucune différence entre cette option et celle de remplissage qui bourre la fin de l'en-tête du datagramme de zéros jusqu'à la limite d'un mot de quatre octets. En raison de son rôle, il est clair que cette option peut être copiée, introduite ou détruite à la fragmentation ou pour toute autre raison (RFC 791, p. 16).

2.5.5.3. *Aucune opération*

Format : octet de type d'option = 00000001

Cette option peut être utilisée entre les options pour aligner le début de l'option qui suit avec le début d'un mot de quatre octets. Elle peut être copiée, introduite ou détruite à la fragmentation comme pour toute autre raison (RFC 791, P. 17).

Les deux options que nous venons de décrire peuvent en raison de leurs fonctions apparaître plusieurs fois dans l'en-tête d'un datagramme; les six options qui suivent ne peuvent par contre s'y présenter qu'une fois au plus. Elles adoptent le second format, celui qui comprend un octet de type d'option, un octet de longueur d'option et un nombre variable d'octets des données de l'option.

2.5.5.4. *Sécurité*

Format : octet de type d'option = 10000010
octet de longueur d'option = 11 (valeur décimale)

Le champ données de l'option (9 octets) regroupe quatre champs se rapportant à la sécurité. Le premier (2 octets) spécifie le niveau de sécurité. Le protocole IP a spécifié huit niveaux de sécurité; il en a réservé huit autres pour un usage futur.
Le deuxième champ (2 octets) spécifie l'appartenance de l'information véhiculée à un département particulier. Une agence du Département de la Défense (The Defense Intelligence Agency) tient à jour un répertoire de ces départements. Les deux derniers champs (5 octets) contiennent les paramètres de codage propres au Département de la Défense. Cette option doit bien sûr être recopiée en cas de fragmentation (RFC 791, pp. 17-18).

2.5.5.5. *Routage source lâche et enregistrement de la route*

Format : octet de type d'option = 10000011
octet de longueur d'option = de valeur variable avec une limite supérieure fixée à 40 octets puisque l'en-tête d'un datagramme internet ne peut compter plus de 60 octets.

Cette option fournit à l'émetteur d'un datagramme la possibilité d'introduire dans ce datagramme des informations qui définissent la route qu'il y a à suivre. Cela prend la forme d'une suite d'adresses internet désignant les passerelles par lesquelles le datagramme doit passer. Ce routage source est lâche, c'est-à-dire que pour atteindre l'adresse internet suivante dans la liste, n'importe quel chemin peut être emprunté. En outre, le chemin effectivement suivi par le datagramme est enregistré par l'insertion de l'adresse internet de la passerelle visitée entre les adresses internet des passerelles déjà visitées et celles qui restent à fréquenter (RFC 791, p. 18). Cette option est recopiée en cas de fragmentation.

2.5.5.6. *Routage source strict et enregistrement de la route*

Format : octet de type d'option = 10001001
octet de longueur d'option = semblable au point précédent.

Cette option est analogue à la précédente à cela près que le routage source est strict, c'est-à-dire que le datagramme sera chaque fois directement envoyé à l'adresse suivante dans la liste des adresses internet à visiter.

2.5.5.7. *Enregistrement de la route*

Format : octet du type d'option = 00000111
octet de longueur d'option = semblable au point précédent.

Cette option permet d'enregistrer dans l'en-tête du datagramme le chemin qu'il suit. En cas de fragmentation, cette option n'est reprise que dans le premier datagramme.

2.5.5.8. *Identificateur "stream"*

Format : octet de type d'option = 10001000
octet de longueur d'option = 4.

Sur base des spécifications du protocole IP, nous n'avons pu dégager la signification de ce champ. Signalons quand même que cette option doit être recopiée dans chaque datagramme en cas de fragmentation.

2.2.5.9. Estampillage

Format : octet de type d'option = 01000100
octet de longueur d'option = de valeur variable avec une limite supérieure fixée à 40 octets.

Cette option permet d'enregistrer le moment où le datagramme est traité à chaque module IP. Le datagramme qui arrive à destination reprendra la liste des couples < adresse, heure > des modules IP visités. En cas de fragmentation, cette option n'est reprise que dans le premier datagramme.

2.5.5.10. *Protocole ISO 8473*

Le projet ISO/DIS 8473 a retenu les sept options suivantes :

- le remplissage,
- la sécurité,
- le routage source lâche,
- le routage source strict,
- l'enregistrement de la qualité de service,
- la priorité.

Le format des options du protocole ISO 8473 est fort comparable à celui du protocole IP. Ce format comprend :

- un champ de code de paramètre (1 octet) qui identifie l'option,
- un champ de longueur de paramètre (1 octet) qui indique la longueur en octets du champ de valeur de paramètre,
- un champ de valeur de paramètre (de longueur variable) qui contient la valeur du paramètre identifié dans le champ de code de paramètre.

2.5.6. Les données

2.5.6.1. *Protocole IP*

La partie des données du datagramme compte un multiple d'octets. Elle transporte les données que l'utilisateur du service datagramme veut transmettre. Sa longueur maximale théorique est de 65 535 octets. Dans la pratique, chaque module IP doit être capable de recevoir un datagramme de 576 octets.

2.5.6.2. *Protocole ISO 8473*

Dans le protocole ISO 8473, on distingue deux types de NPDU : les NPDU de données et les NPDU de rapport d'erreur. Les premières sont comparables aux datagrammes du protocole IP; leur partie données a également une structure d'un multiple ordonné d'octets. Les secondes sont analogues aux messages ICMP; leur partie données est utilisée pour renseigner l'émetteur d'une NPDU de la raison du rejet d'une NPDU de données. Ce type de NPDU sera présenté au troisième chapitre consacré au protocole ICMP.

2.6. Résumé

Nous voudrions résumer ici les enseignements les plus importants que cette étude des protocoles IP et ISO 8473 nous a apportés. Nous allons d'abord présenter les différences marquantes entre les deux protocoles :

- au niveau de la décomposition en couches, le protocole IP regroupe les fonctions des protocoles SNICP et SNDCP du modèle ISO alors que le protocole ISO 8473 est typiquement un protocole SNICP;

- au niveau de l'adressage, l'adresse internet est de longueur fixe tandis que les informations NPAI sont de longueur variable; de plus, l'adresse internet contient des informations de routage alors que les informations NPAI ont été conçues pour ne pas influencer le chemin que va utiliser le fournisseur de service.

Les points de convergence entre les deux protocoles sont beaucoup plus nombreux; nous allons seulement mentionner ici les plus importants :

- au niveau du service, ils utilisent tous les deux un service réseau datagramme et ils offrent un service de transport bout en bout non fiable;

- au niveau de l'adressage, ils distinguent deux types d'adresse; d'une part, les adresses internet et les informations NPAI, et d'autre part, les adresses sous-réseau. La fonction de mise à jour des tables de routage est laissée à des protocoles spécifiques; les protocoles IGP et EGP pour le modèle DoD;

- au niveau des unités de transmission, le protocole ISO 8473 distingue les NPDU de données et les NPDU de rapport d'erreur qui correspondent respectivement aux datagrammes internet du protocole IP et aux messages ICMP.

CHAPITRE 3

LE PROTOCOLE ICMP

3.1. Introduction

Le protocole ICMP est un partenaire obligé du protocole IP. Placé généralement au même niveau que le protocole IP, il utilise les services fournis par IP pour transmettre ses messages.

L'intérêt de ce protocole est d'abord de fournir aux passerelles et aux ordinateurs hôtes destinations la possibilité de renseigner les ordinateurs hôtes sources des erreurs détectées lors du traitement des datagrammes. Ce protocole propose également un certain nombre de mécanismes en vue

- de signaler l'arrivée de situation de congestion,
- de tenir à jour les tables de routage et
- de récolter des informations relatives aux performances du catenet.

Ce protocole n'a pas pour but de rendre le protocole IP plus fiable. En effet, l'ordinateur hôte source n'a toujours aucune garantie que le datagramme envoyé est bien arrivé à destination même si aucun message ICMP ne lui est arrivé. Ce manque de fiabilité est renforcé par les deux règles suivantes :

- aucun message ICMP n'est envoyé quand on détecte une erreur dans un autre message ICMP;

- un message ICMP n'est envoyé que pour les erreurs détectées dans les datagrammes dont le champ *décalage* est égal à zéro; rappelons que ce mécanisme[1] sert à indiquer que le datagramme transporte le premier octet du champ de *données* du datagramme initial.

Nous avons déjà signalé[2] que le protocole ISO 8473 dispose d'une fonction fort analogue à celles du protocole ICMP. Il s'agit de la fonction de rapport d'erreur.

Nous allons organiser notre étude du protocole ICMP et de cette fonction de la manière suivante : à la section 3.2., nous présenterons le format commun des messages ICMP ainsi que les onze types de message ICMP; à la section 3.3., nous décrirons la fonction de rapport d'erreur du protocole ISO 8473.

---

1 Cfr. paragraphe 2.4.2. pour l'explication de ce mécanisme.
2 Cfr. paragraphe 2.5.6.1.

3.2. Format et types de messages ICMP

Le module ICMP est activé chaque fois que le module IP détecte dans un datagramme une erreur pour laquelle un message ICMP a été prévu. Le module ICMP construit alors ce message. Il est ensuite passé au module IP qui l'encapsule dans un datagramme internet. Après cela, le datagramme est envoyé à l'ordinateur hôte source du datagramme erroné.

Les messages ICMP partagent un format commun. Ce format comprend les quatre champs suivants :

- un champ *type de message* (1 octet) qui spécifie le type de message ICMP; il définit également le format du reste du message et la signification du champ *code* ;
- un champ *code* (1 octet) qui spécifie comment les champs *paramètres* et *information* doivent être interprêtés;
- un champ *somme de contrôle* (2 octets) que l'ordinateur hôte source utilise pour vérifier l'intégrité du message ICMP;
- les champs *paramètres* (4 octets) et *information* (de longueur variable) dont la signification dépend des champs *type de message* et *code* .

Le protocole ICMP distingue onze types de message ICMP que nous allons maintenant passer en revue.

3.2.1. Destination hors de portée

Format : octet de type de message = 3.

Ce message recouvre six situations pour lesquelles le champ *code* prend les valeurs suivantes :

- *code* = O; ce message est envoyé lorsqu'une passerelle ne sait pas comment atteindre le réseau destination ou lorsque le réseau destination est inactif;
- *code* = 1; ce message est envoyé lorsqu'une passerelle ne sait pas atteindre l'ordinateur hôte destination parce que celui-ci n'existe pas ou parce qu'il est inactif;
- *code* = 2; ce message est envoyé par l'ordinateur hôte destination lorsque personne n'a demandé à recevoir ce datagramme;

- *code* = 3; ce message est envoyé par l'ordinateur hôte destination lorsqu'il n'existe aucun processus attaché à la porte référencée par le champ *porte destination* [3] contenu dans l'en-tête du segment TCP ou dans celle du datagramme UDP (encapsulé dans le datagramme);

- *code* = 4; ce message est envoyé par une passerelle qui devrait fragmenter un datagramme dont le drapeau DF est mis à 1;

- *code* = 5; ce message est envoyé par une passerelle lorsque le routage source spécifié dans le datagramme ne peut être concrétisé.

Pour ces six messages, le champ *paramètres* n'existe pas tandis que le champ *information* reprend l'en-tête complet du datagramme détruit plus les 64 premiers bits du champ de *données* de ce datagramme. Ces derniers renseignements sont très importants parce qu'ils permettent de retrouver le processus qui a envoyé le datagramme.

3.2.2. Temps dépassé

Format : octet de type de message = 11.

Le champ *code* peut prendre l'une des deux valeurs suivantes :

- *code* = 0; ce message est émis par une passerelle lorsque le champ *durée de vie* du datagramme est devenu nul;

- *code* = 1; ce message est émis par l'ordinateur hôte destination lorsque le délai imparti pour le réassemblage du datagramme initial est épuisé.

Le message ne comprend toujours aucun champ *paramètres* tandis qu'il reprend le même champ *information* que celui décrit pour le premier type de message.

3.2.3. Problème de paramètres

Format : octet de type de message = 12,
octet de code = 0.

Ce message est envoyé lorsqu'une passerelle ou l'ordinateur hôte destination a détecté un problème de paramètrage dans l'en-tête du datagramme. Le champ *paramètres* contient alors un pointeur vers l'octet de l'en-tête du datagramme où l'erreur a été détectée. Le champ *information* a un contenu identique à celui décrit pour les deux premiers types de message ICMP; en particulier, il reprendra intégralement l'en-tête du datagramme erroné.

3 Cfr paragraphe 4.2.2. pour la signification de ce concept.

3.2.4. Congestion

Format : octet de type de message = 4,
octet de code = 0.

Ce message[4] peut être émis si l'une des trois situations suivantes se produit :

- une passerelle peut détruire des datagrammes si elle n'a plus l'espace mémoire suffisant pour les stocker. Chaque fois qu'elle en détruira un, elle sera tenue d'envoyer un message de congestion à l'ordinateur hôte source du datagramme. Celui-ci pourrait réagir en diminuant la fréquence avec laquelle il émet des datagrammes vers cette destination;

- l'ordinateur hôte destination peut estimer que les datagrammes lui arrivent trop vite pour pouvoir être traités; il enverra alors un message de congestion à l'ordinateur hôte source;

- une passerelle ou un ordinateur hôte peuvent éviter une situation de congestion en l'anticipant par l'envoi de ce message une fois un certain niveau de capacité utilisé. Dans ces deux derniers cas, la spécification du protocole ICMP ne précise pas si cela provoque la destruction d'un datagramme.

A la réception d'un tel message, une réaction possible de l'ordinateur hôte source sera de diminuer la fréquence d'émission des datagrammes vers cette destination. A partir du moment où il ne reçoit plus de tels messages, il pourra graduellement réaugmenter sa fréquence jusqu'à ce qu'il reçoive éventuellement à nouveau ce message.

Le champ *information* a un contenu identique à ceux des trois premiers types de message ICMP. Il n'existe également aucun champ *paramètres*.

3.2.5. Message de redirection

Format : octet de type de message = 5.

Situons brièvement le problème. Un datagramme vient d'être émis par l'ordinateur hôte source S. Il est adressé à l'ordinateur hôte destination D. Suite à une décision de routage de S, le datagramme transite d'abord par la passerelle P1. S et P1 sont donc bien connectés au même réseau. P1 consulte sa table de routage et décide d'envoyer le datagramme à la passerelle P2. Si P2 et S sont reliés au même réseau, un message de redirection est alors envoyé à S. La figure 3.1. représente cette situation.

---

4 Cfr. paragraphe 4.3.3.3. pour un exemple d'utilisation de ce type de message.

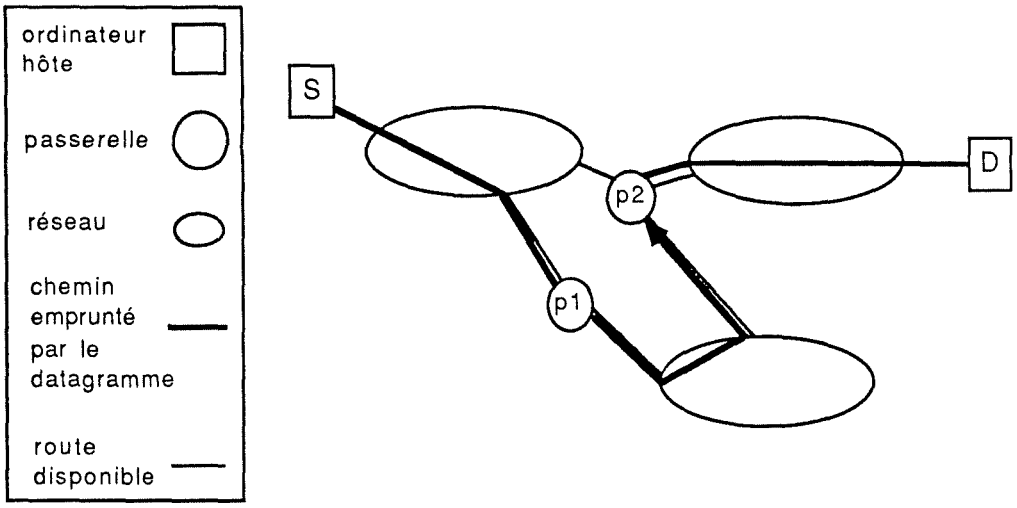


Fig. 3.1. - Exemple d'utilisation d'un message de redirection.

Ce message de REDIRECTION n'est envoyé que si la situation décrite ici plus haut se produit et si l'option de routage source n'est pas sélectionnée. Ce message indique à l'ordinateur hôte source qu'il existe un chemin plus court pour accéder au réseau auquel l'ordinateur hôte D est relié. Ce message n'a qu'une valeur informative et donc la mise à jour de la table de routage de S ne sera pas automatique.

Le champ *paramètres* reprend l'adresse de la passerelle qui a mis en évidence la situation; dans notre exemple, il s'agit de P2. Cette adresse sera celle que la fonction de routage du module IP de S pourra sélectionner quand il aura à adresser un datagramme à l'un des ordinateurs hôtes attachés au même réseau que D. Le champ *information* est identique aux quatre messages précédents.

3.2.6. Echo et retour d'écho

Pour les messages d'ECHO, le format est le suivant :
octet de type de message = 8,
octet de code = 0.

Pour les mesages de RETOUR D'ECHO, le format est le suivant :
octet de type de message = 9,
octet de code = 0.

Le message d'ECHO est envoyé par un ordinateur hôte qui veut forcer un ordinateur hôte à lui répondre en lui renvoyant le message reçu dans un message de type RETOUR D'ECHO. Ce mécanisme permet de tester la communication entre deux ordinateurs hôtes, de vérifier qu'un ordinateur hôte est encore actif ou d'obtenir des renseignements statistiques sur le fonctionnement du catenet.

Le champ *paramètres* comprend deux identificateurs longs de deux octets chacun. L'émetteur du message ECHO gère ces identificateurs comme il l'entend. Le champ *information* contient des données définies par l'émetteur du message ECHO et qui seront renvoyées dans ce champ du message de RETOUR D'ECHO.

3.2.7. Estampille et retour d'estampille

Pour les messages d'ESTAMPILLE, le format est le suivant :
octet de type de message = 13,
octet de code = 0.

Pour les messages de RETOUR D'ESTAMPILLE, le format est le suivant :
octet de type de message = 14,
octet de code = 0.

Ces deux messages sont analogues aux deux précédents mais ce sont les repères de temps qui sont pris en compte par l'émetteur et le récepteur de ces messages.

Le champ *paramètres* comprend également deux identificateurs de deux octets chacun dont l'émetteur du message d'ESTAMPILLE dispose comme il l'entend. Le champ *information* inclut à l'émission du message le dernier moment où il est encore sous le contrôle de l'émetteur avant d'être envoyé. Dès que le message arrive au récepteur, on ajoute dans le champ *information* le moment précis où il entre sous son contrôle. Enfin, on joint encore à ces deux estampilles le moment où le message de RETOUR D'ESTAMPILLE est renvoyé.

3.2.8. Demande d'information et réponse à la demande

Pour les messages de DEMANDE D'INFORMATION, le format est le suivant :
octet de type de message = 15,
octet de code = 0.

Pour les messages de REPONSE A LA DEMANDE, le format est le suivant :
octet de type de message = 16,
octet de code = 0.

Ces deux messages permettent à un ordinateur hôte de connaître le numéro du réseau auquel il est connecté. Pour envoyer ce message, il se pose évidemment le problème de la détermination du numéro de réseau dans les adresses internet (source et destination) contenues dans l'en-tête du datagramme internet puisque ce numéro constitue précisément l'objet de la demande. La solution retenue est de définir par convention le numéro de réseau zéro comme étant le numéro du réseau auquel l'ordinateur hôte est connecté. La réponse à la requête se trouvera en fait dans la partie numéro de réseau des adresses internet reprises dans l'en-tête du datagramme qui transporte le message de REPONSE A LA DEMANDE. Le choix de l'interlocuteur n'est pas spécifié par le protocole ICMP. On peut toutefois supposer que l'ordinateur hôte n'envoyera un message DEMANDE D'INFORMATION qu'à une passerelle ou à un ordinateur hôte dont une de ses fonctions est de fournir ce type de renseignement.

Le champ *paramètres* de ces deux messages comprendra deux identificateurs définis par l'émetteur de la demande tandis qu'aucun champ *information* n'est prévu.

3.3. La fonction de rapport d'erreur dans le protocole ISO 8473

L'objet de la fonction de rapport d'erreur du protocole ISO 8473 est la communication d'une NPDU de rapport d'erreur à l'entité source d'une NPDU de données quand cette NPDU est rejetée. Cette fonction se démarque du protocole ICMP sur les deux points suivants :

- la présence d'un drapeau de rapport d'erreur placé par l'entité-source de réseau dans la NPDU initiale pour indiquer qu'une NPDU de rapport d'erreur doit être retournée en cas de rejet de la NPDU. Ce caractère optionnel de la fonction n'existe pas dans le protocole ICMP puisqu'un message ICMP doit être renvoyé chaque fois qu'une erreur est détectée;

- la fonction de rapport d'erreur n'est activée que si la NPDU est rejetée; elle ne prévoit donc aucun mécanisme de routage ou d'échange d'informations relatives au fonctionnement du réseau.

Nous avons choisi de ne pas mentionner comme différence le fait que la fonction de rapport d'erreur est un élément du protocole ISO 8473 alors qu'elle est implémentée en dehors du protocole IP dans l'architecture DoD. Ce choix s'explique pour les deux raisons suivantes :

- le protocole ICMP doit être implémentée avec chaque module IP; il est donc une partie intégrante du protocole IP;

- comme dans le protocole ICMP, les informations relatives au rapport d'erreur sont encapsulées dans une NPDU. Rappelons que pour distinguer une NPDU de données d'une NPDU de rapport d'erreur, on utilise le champ *code de type* (5 bits) qui, s'il prend la valeur 11100, identifie une NPDU de données et qui, s'il prend la valeur de 00001, identifie une NPDU de rapport d'erreur.

Les remarques relatives au manque de fiabilité du protocole ICMP qui ont été faites en introduction restent valables pour la fonction de rapport d'erreur et, en particulier, la non-réception d'une NPDU de rapport d'erreur n'implique pas une transmission correcte de la NPDU de données.

Les informations spécifiques au rapport d'erreur sont regroupées dans les deux champs suivants :

- le champ *motif de rejet* (4 octets) et
- le champ de *données* du rapport d'erreur (de longueur variable).

Le champ *motif de rejet* reprend la même structure que celle définie pour les options, ce qui renforce le caractère optionnel de la fonction. On retrouve donc :

- un octet de code de paramètre = 11000001,
- un octet de longueur du paramètre = 2,
- un octet de type d'erreur,
- un octet de référence.

L'octet de type d'erreur identifie le type d'erreur rencontré. Il correspond aux champs *type de message* et *code* des messages ICMP. L'octet de référence contient un pointeur visant le champ (contenu dans le champ de données de rapport d'erreur) de la NPDU rejetée qui a déclenché l'erreur. Si l'erreur ne peut être associé à aucun champ particulier, cet octet prendra la valeur zéro. Le champ de *données* du rapport d'erreur contient au moins l'en-tête de la NPDU rejetée. Grâce aux informations contenues dans ces deux champs, l'entité-source peut identifier la NPDU rejetée ainsi que l'octet qui est à l'origine du rejet.

Une NPDU de rapport d'erreur doit être générée

- si le drapeau de rapport d'erreur de la NPDU rejetée autorise l'envoi d'un tel rapport;
- si le motif de rejet est l'un des suivants :
    - une violation du protocole,
    - la somme de contrôle incorrecte,
    - le rejet pour cause d'encombrement,
    - une erreur de syntaxe à l'en-tête,
    - une segmentation nécessaire mais interdite,
    - la destination hors de portée ou inconnue,
    - le routage source incorrect, il peut s'agir d'une erreur de syntaxe au niveau du champ de routage source, de la mention d'une adresse inconnue ou hors de portée, ou d'un itinéraire inacceptable pour d'autres raisons,
    - la durée de vie expirée pendant le transit de la NPDU ou au cours du réassemblage,
    - une option non prise en charge.

Une NPDU de rapport d'erreur peut être générée, si une NPDU dont le drapeau autorise les rapports d'erreur est rejetée pour tout autre motif.

De cette présentation, il se dégage que la fonction de rapport d'erreur du protocole ISO 8473 ne diffère pas vraiment du protocole ICMP. A chaque type de NPDU de rapport d'erreur, on peut en effet faire correspondre un type de message ICMP équivalent, excepté dans le cas d'une somme de contrôle incorrecte. Dans ce cas, le protocole ISO 8473 prévoit l'envoi d'une NPDU de rapport d'erreur tandis que le protocole ICMP ne le prévoit pas. On peut supposer que ce choix du modèle DoD se justifie par la difficulté de savoir si le champ *adresse source* du datagramme a été, oui ou non, endommagé.

CHAPITRE 4

LE PROTOCOLE TCP

4.1. Introduction

Au niveau bout en bout, le modèle DoD a spécifié deux protocoles : il s'agit des protocoles TCP et UDP. Ces deux protocoles ont été créés pour fournir un service de communication entre processus appartenant à des ordinateurs hôtes distincts, connectés à des réseaux éventuellement différents mais interconnectés. Pour le protocole TCP, ce service est fiable, c'est-à-dire que tous les messages envoyés par un processus arrivent corrects, dans l'ordre et sans duplication à l'autre processus[1]. On dit alors que ce service est orienté circuit virtuel. A l'opposé, le protocole UDP offre un service non fiable, orienté datagramme. Ces deux protocoles supposent qu'ils disposent de protocoles de niveau inférieur qui leur fournissent un service datagramme non fiable tel que le protocole IP le fournit. Pour offrir ce service fiable, le protocole TCP utilise des mécanismes complexes de multiplexage, de gestion de connexion , de transfert de données, de contrôle de flux, de priorité et de sécurité.

L'objet de ce chapitre n'est pas de présenter une spécification complète du protocole TCP. Nous allons plutôt nous limiter à décrire les mécanismes les plus importants attachés à ce protocole. Nous présenterons également le principal problème rencontré dans l'architecture DoD. Il s'agit du phénomène de congestion. Des études récentes (Magle 84, Rose 85, Zhang 87) tentent en effet à prouver qu'une architecture basée, comme le modèle DoD le propose, sur un service datagramme au niveau inter-réseaux et sur un service circuit virtuel au niveau bout en bout, est une des causes majeures du problème de congestion rencontré sur les catenets.

---

[1] Cette affirmation sera nuancée au paragraphe 4.2.4.

4.2. Principaux mécanismes utilisés par le protocole TCP

4.2.1. Introduction

Dans chaque ordinateur hôte, le protocole TCP est implémenté et géré par un **module TCP**. Ce module est une partie importante du système d'exploitation puisqu'il gère l'allocation du système de communication entre les différents processus applications. L'interface aux processus applications comprend un certain nombre de fonctions qui correspondent en fait à des appels systèmes.

Pour donner une vue globale et succincte du fonctionnement du protocole TCP, nous décrirons d'abord les mécanismes du multiplexage; ensuite, nous présenterons comment une connexion est établie, comment les données sont échangées à travers cette connexion et comment cette connexion est clôturée; enfin, nous décrirons les mécanismes de priorité et de sécurité du protocole TCP.

4.2.2. Multiplexage

Pour permettre à plusieurs processus applications d'utiliser les services offerts par le protocole TCP, le module TCP de chaque ordinateur hôte gère un ensemble de **portes**. C'est à partir de ces portes que chaque processus utilisateur pourra utiliser les services de communication du protocole TCP. Si on concatène à ce numéro de porte, l'adresse internet de l'ordinateur hôte local, on forme une **adresse transport** qui est également appelée **socket**. Cette adresse transport permet d'identifier un seul processus application à travers tout le catenet. Une paire de ces adresses définit de manière unique une **connexion**. Un socket peut être utilisé simultanément dans plusieurs connexions. Une communication entre deux processus peut impliquer différentes connexions de telle sorte que différents numéros de porte peuvent être attribués à un même processus application.

Pour les processus qui sont liés à des applications utilisés fréquemment, il est convenu que ceux-ci recevront des numéros de porte identiques pour une application donnée, quel que soit l'ordinateur hôte. Ce numéro est strictement réservé à l'application. Tous ces numéros de porte "**bien connus**" ont été attribués de manière définitive. Pour les autres processus moins connus, l'attribution de leurs numéros implique des mécanismes dynamiques.

4.2.3. Etablissement d'une connexion

Si un processus application veut échanger des informations avec un processus application situé sur un autre ordinateur hôte, ces processus devront d'abord établir entre eux une connexion. Cette phase d'établissement de connexion permet aux deux processus de s'assurer de leur existence mutuelle, de négocier un certain nombre de paramètres et d'allouer les ressources requises pour la connexion. Toutes les informations relatives à une connexion particulièree sont conservées dans un enregistrement appelé **bloc de contrôle de transmission** (bloc TCB, pour Transmission Control Block). Il existe un bloc TCB à l'extrémité de chaque connexion. Cet enregistrement contient, entre autres, les deux adresses de transport et des variables qui décrivent l'état local de la connexion.

Pour établir une connexion, le processus application appelle la fonction OPEN implémentée dans le module TCP. Les arguments transmis lors de cet appel sont au moins le numéro de porte local, l'adresse transport du processus avec lequel il veut entrer en contact et une indication pour signaler que la connexion est passive ou active. Si le module TCP doit établir une **connexion active**, il doit commencer une procédure d'établissement de connexion en trois étapes. Une telle procédure échange trois segments pour créer une connexion et plus particulièrement pour synchroniser les deux modules TCP qui participent à la connexion. Dans le cas d'une **connexion passive**, le module TCP s'apprêtera à recevoir des demandes d'établissement de connexion.

L'adresse transport du processus éloigné qui est fournie comme argument lors de l'appel OPEN[2] peut être complètement spécifiée ou pas du tout. Elle n'est pas spécifiée lorsqu'elle n'est composée que de zéros. Une adresse transport non spécifiée n'est permise que lors de l'établissement de connexion passive. Un processus qui rend des services demande généralement d'ouvrir une telle connexion. Il aura alors un numéro de porte bien connue de tous.

Deux appels à la fonction OPEN pour ouvrir une connexion active mais aussi un appel pour ouvrir une connexion passive et un autre pour ouvrir une connexion active permettent d'établir une connexion à la condition que les portes passées dans les appels correspondent.

---

2 Cfr. paragraphe 5.4.1. pour la spécification de cette fonction.

4.2.4. Transfert de données

Une fois la connexion établie, chaque processus peut envoyer et recevoir simultanément des flots continus d'octets[3]. Ces flots envoyés par un module TCP seront délivrés corrects, dans l'ordre et sans duplication à l'autre extrémité de la connexion. Pour transférer des données à travers une connexion établie, un module TCP va d'abord les découper; il va ensuite les envelopper dans des segments TCP et enfin, confier au module IP la charge de les transmettre sur le catenet. Un **segment TCP** est l'unité de transmission de données échangée entre deux modules TCP.

En général, les modules TCP décident de bloquer ou de transférer les données à leur propre convenance. Parfois, les processus applications ont besoin d'être sûrs que toutes les données qui ont été soumises au module TCP ont bien été transmises. Dans ce but, lors de l'appel de la fonction SEND[4] pour envoyer des données, le processus application sélectionnera le **drapeau PUSH**; cette sélection va assurer le processus que toutes les données qu'il a fournies au module TCP jusqu'au moment de l'appel, vont être immédiatement envoyées au destinataire. Remarquons bien que cette option PUSH reste invisible au processus application qui reçoit les données.

La transmission des données est rendue fiable par l'utilisation de **numéro de séquence** et d'**accusé de réception**. Le mécanisme de base est d'attacher à chaque octet des données un numéro de séquence. Le numéro de séquence du premier octet contenu dans le champ de *données* du segment est transmis avec ce segment. Les segments transmettent également un numéro d'accusé de réception qui indique quel sera le numéro de séquence du prochain octet de données qui pourra être envoyé en sens inverse.

Quand le module TCP doit transmettre un segment contenant des données, il place une copie dans une file de retransmission en y associant l'heure de départ. Quand un accusé de réception arrive, le segment qui y est associé est détruit de la file de retransmission. Si aucun accusé n'arrive dans un laps de temps fixé, le segment sera retransmis.

Quand le module TCP reçoit un accusé de réception, cela ne garantit pas que le processus application a bien reçu les données; cela signifie seulement que le module TCP destinataire a pris la responsabilité de les transmettre au processus application approprié. Les numéros de séquence sont utilisés pour réordonner les segments reçus et éliminer les doublures. Les segments endommagés lors de la transmission à travers le catenet sont détectés grâce au mécanisme de la **somme de contrôle**. La somme de contrôle est calculée sur l'en-tête du segment et son champ de *données* .

---

[3] Le protocole TCP ne permet donc pas le transfert en séquence d'enregistrements (Record Oriented Service).

[4] Cfr. paragraphe 5.4.2. pour la spécification de cette fonction.

Pour contrôler le flot des données échangées entre les deux modules TCP, le protocole TCP utilise un mécanisme qui permet à un module TCP de règler la quantité de données émises par l'autre module TCP. Ce mécanisme est celui de la fenêtre. Une **fenêtre** indique le nombre d'octets à partir des données déjà acquittées que le module TCP est prêt à recevoir. On voit directement que le débit maximum est limité par le délai de transit[5] des datagrammes sur le catenet puisque le rythme d'émission dépend du délai de réception des acquittements.

4.2.5. Fermeture d'une connexion

Etant donné qu'une connexion permet d'échanger simultanément des données dans les deux sens, la clôture d'une connexion par une extrémité est relativement délicate. Quand un processus application estime qu'il n'a plus de données à recevoir de l'autre extrémite, il peut demander au module TCP de fermer la connexion en appelant la fonction CLOSE[6]. Le processus application pourra toutefois continuer à recevoir des données de l'autre extrémité de la connexion jusqu'à ce que cette dernière, renseignée de la demande de fermeture émanant de son interlocuteur, décide à son tour de clôturer la connexion.

4.2.6. Priorité et sécurité

Au moment de l'établissement d'une connexion, les processus applications peuvent indiquer au module TCP des niveaux de priorité et de sécurité pour leur communication. Ces aspects ne sont pas implémentés par le protocole TCP mais sont pris en charge par le protocole IP. Rappelons-nous que le champ *type de service* du datagramme internet permet d'indiquer le niveau de priorité du datagramme et qu'il existe en option avec le protocole IP un mécanisme de sécurité.

---

[5] Ce délai est le délai nécessaire pour envoyer les données à destination et recevoir leur acquittement.

[6] Cfr. paragraphe 5.4.4. pour la spécification de la fonction CLOSE.

4.3. Le problème du contrôle de congestion

4.3.1. Présentation du problème

"*Tous les réseaux de communication tels que réseau routier, réseau à commutation par paquets ou réseau à commutation de circuits disposent de ressources limitées pour assurer l'acheminement qui leur incombe. Pour un réseau à commutation par paquets, ces limitations se situent principalement au niveau du débit maximum des lignes, de la capacité de traitement des noeuds et de la dimension des mémoires tampons qui implantent les files d'attente émission et réception des voies de communication. Lorsque la demande de trafic croît, le réseau subit des phénomènes de* ***congestion*** *qui se traduisent, si on ne prend pas de précautions spéciales, par un effondrement des performances du réseau* " (Nussbaumer 87, p. 117).

"*Les performances d'un réseau peuvent être caractérisées par différents critères tels que le trafic utile écoulé par le réseau, le temps de transfert moyen des paquets, le nombre de paquets perdus ou en erreur. Si nous prenons par exemple en considération le critère du trafic utile, c'est-à-dire le nombre de paquets délivrés par unité de temps aux utilisateurs, un réseau idéal devrait être capable d'acheminer un trafic utile directement proportionnel au trafic offert au réseau, et ce jusqu'à ce que la capacité maximum de transport du réseau soit atteinte. Au delà de cette limite, le réseau devrait être capable de fonctionner à capacité maximum, quel que soit le trafic offert.*" Ce comportement est représenté par la courbe (a) de la figure 4.1. qui illustre le phénomène de congestion en utilisant le critère du trafic écoulé. "*En pratique, le fonctionnement du réseau s'écarte de l'idéal pour un certain nombre de raisons qui ont toutes trait à une allocation inefficace des ressources en cas de surcharge. En particulier, la dimension des files d'attente sur les lignes croît rapidement en fonction du trafic et arrive à dépasser la capacité des mémoires tampons des noeuds à l'approche de la capacité maximum. Les paquets qui ne peuvent plus être stockés en mémoire doivent alors être jetés, ce qui provoque leur retransmission et donc propage l'engorgement vers les liaisons situées en amont en provoquant l'apparition d'un trafic interne supplémentaire. Il y a donc un phénomène d'avalanche dû au fait qu'au-delà d'une certaine limite, la capacité utile de transport du réseau diminue lorsque le trafic offert croît*" (Nussbaumer 87, p. 117) . La courbe (b) de la figure 4.1. représente ce phénomène de congestion.

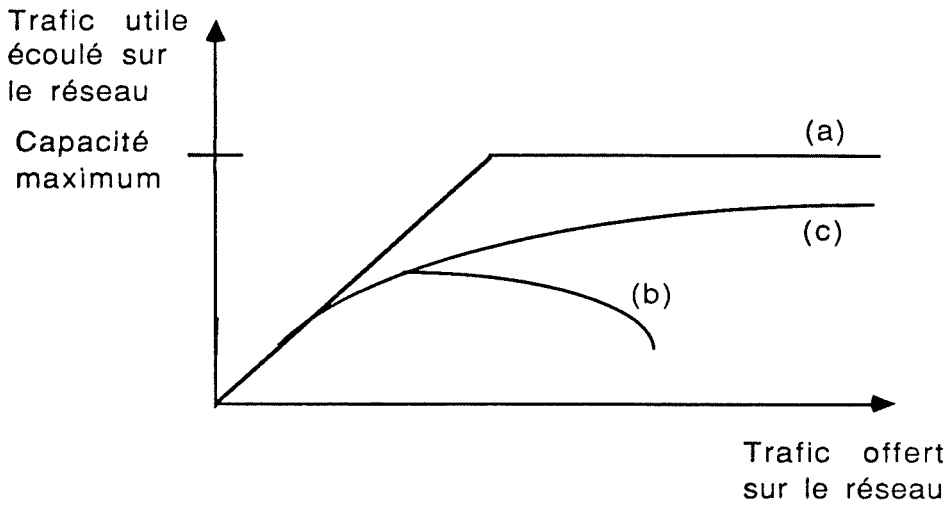


Fig. 4.1. - Phénomène de congestion : (a) comportement idéal; (b) congestion; (c) performances d'un réseau avec contrôle de congestion.

A côté des limitations spécifiques à chaque réseau, apparaissent avec un catenet une série de nouvelles causes de congestion. Ces causes se situent généralement au niveau des différences entre les débits des réseaux connectés, au niveau de la capacité de traitement des passerelles et au niveau de l'espace mémoire des passerelles. Mais contrairement à ce qui se passe dans un réseau, le phénomène de congestion est plus ardu à cerner dans un catenet par le fait que, d'une part, les réseaux connectés ont des propriétés différentes et que, d'autre part, les passerelles sont rarement identiques. Ainsi, la largeur de bande des liaisons de données de ces réseaux peut varier de 1200 à 10000000 bits par seconde et les services offerts par ces réseaux peuvent être très différents. En outre, les passerelles peuvent remplir d'autres fonctions que celles de relais et de routage. L'exécution de ces autres fonctions peut également entraîner la déterioration des performances de la passerelle.

Nous allons organiser le reste de cette section autour du problème de congestion observé dans l'architecture DoD. Au paragraphe 4.3.2., nous dresserons la liste des raisons actuellement invoquées pour expliquer le problème de congestion dans le modèle DoD. Au paragraphe 4.3.3., nous présenterons deux mécanismes de contrôle de congestion proposés comme solution à ce problème. La courbe (c) de la figure 4.1. illustre le résultat escompté par l'emploi de tels mécanismes.

4.3.2. Causes de congestion spécifiques au modèle DoD

Rappelons-nous que le modèle DoD présente la caractéristique d'offrir avec le protocole TCP un service de communication parfait aux processus applications en se basant sur un service de transmission simple, flexible mais peu fiable. Le protocole IP qui offre ce service, délivre les datagrammes comme des entités indépendantes. Aucune information concernant les applications qui utilisent les services du protocole TCP n'est conservée par les passerelles à l'intérieur du catenet.

Ce type de système d'interconnexion présente des propriétés très intéressantes : la plus importante est sans doute de fournir un service de transmission plus fiable en cas de défaillance d'un des composants du catenet tels que passerelle, ligne ou réseau entier. Les datagrammes peuvent alors assez facilement changer de parcours pour éviter la partie défectueuse du catenet. Un autre avantage est de simplifier la conception des passerelles et de faciliter l'interconnexion de réseaux hérérogènes. Enfin, il laisse aux concepteurs du modèle DoD une grande marge de manoeuvre pour définir de nouveaux protocoles de transmission au niveau bout en bout.

Malheureusement, un tel système d'interconnexion présente un inconvénient majeur : il n'offre aucun mécanisme pour assurer le contrôle du trafic à l'intérieur du catenet. Ceci entraîne de très sérieux problèmes de congestion parce que le catenet ne voit les datagrammes qu'isolément. Il ne reconnait pas les applications qui utilisent les services rendus par le protocole IP.

Quand le catenet doit faire face à une situation de congestion, ou bien certains ordinateurs hôtes vont réagir en refusant d'envoyer des paquets, ou bien certaines passerelles vont commencer à rejeter arbitrairement certains paquets qui provoquent la congestion. Ces deux réactions vont sanctionner de la même manière les applications qui provoquent la situation de congestion que celles qui ne la provoquent.

Ces réactions vont à leur tour entraîner d'autres réactions au niveau du protocole TCP ou au niveau des protocoles applications. Ainsi, le protocole TCP, pour rendre son service transport fiable, va utiliser des techniques de retransmission une fois un certain laps de temps écoulé. Si la première réaction du catenet avait pu provoquer une amélioration temporaire, celle-ci va être vite suivie d'une rechute plus grave encore qui provient du trafic de retransmission. Tant qu'il n'y a qu'une seule copie de chaque datagramme en circulation dans le catenet, la situation de congestion est dominée. Mais, dès que plusieurs copies du même datagramme commence à circuler, le risque de congestion augmente sérieusement.

Normalement, les modules TCP vont essayer de retransmettre leurs segments à plusieurs reprises à intervalles de plus en plus longs. Le nombre de retransmissions ne peut toutefois excèder une limite maximale fixée. Ce mécanisme est suffisant la plupart du temps pour écarter tout danger de congestion. Malheureusement, il peut arriver que la situation soit telle que le module TCP soit amené à clôturer précipitamment la connexion.

4.3.3. Mécanismes de contrôle de congestion

4.3.3.1 *Situation du problème*

Pour éviter que de pareilles situations ne se produisent, la solution la plus simple consiste évidemment à surdimensionner les équipements afin de se placer dans une zone de fonctionnement éloignée de la situation de congestion. Cette solution n'est généralement pas adoptée pour des raisons évidentes de coût. Il faut alors faire appel à un certain nombre de mesures préventives qui sont destinées à assurer un fonctionnement proche de l'idéal à fort trafic. La courbe (c) de la figure 4.1. représente un tel fonctionnement.

Le principal mécanisme prévu dans l'architecture DoD pour prévenir le problème de congestion est celui de la fenêtre coulissante. Ce mécanisme permet de réguler le flux de bout en bout, c'est-à-dire au niveau des ordinateurs hôtes uniquement. Il est basé sur l'idée qu'il faut limiter le nombre de datagrammes en transit dans le catenet afin d'éviter une saturation des ressources. La méthode de la fenêtre coulissante présente toutefois les deux inconvénients suivants : d'une part, elle est indépendante de la structure du catenet et d'autre part, elle n'agit pas directement au niveau du protocole IP. En fait, son effet sur la régulation et la répartition de la charge ne se produit que par ricochet.

Le problème de la congestion est reconnu par la communauté DoD comme l'un des plus importants auxquels l'architecture DoD est confrontée. Depuis 1984, de nombreuses solutions ont été proposées (Nagle 84, Rose 85, Zhang 87). Nous allons en présenter deux proposées par Nagle.

4.3.3.2. *Le problème des petits paquets*

La première proposition (Nagle 84, p. 13-14) cherche au départ à résoudre le problème des petits paquets rencontrés dans les applications de terminal virtuel (protocole Telnet). Cette proposition consiste à empêcher l'émission de nouveaux segments TCP de données tant que toutes les données qui ont déjà été transmises sur la connexion n'ont pas été acquittées. Cette interdiction est inconditionnelle.

Pour le problème de congestion, cette proposition est une mesure préventive puisque les paquets vont normalement transporter plus de données, ce qui va réduire la charge pesant sur le réseau.

Cette proposition peut être généralisée facilement à toutes les applications qui utilisent le service de transport de TCP. Nous allons à l'aide d'un exemple montrer que cette solution ne modifie pas profondément le comportement du protocole TCP pour une autre application que celle du terminal virtuel.

Quand un processus application de transfert de fichier envoit sur une connexion son premier bloc de données, le module TCP qui gère la connexion va l'envoyer immédiatement à l'autre extrémité de la connexion puisqu'aucune donnée n'est encore en attente. On va ensuite attendre que le premier acquittement revienne. Si on suppose que le délai de transit, c'est-à-dire le délai pour l'aller des données et pour le retour de l'acquittement, est de 5 secondes pour la destination choisie, seulement 512 octets de données auront été transmises durant les 5 premières secondes. Ces 512 octets représentent la taille minimum convenue dans le modèle DoD pour le champ des *données* réservées aux protocoles applications. Pendant ces 5 premières secondes, la fenêtre aura normalement été garnie de données et conservée remplie dans le module TCP. Si on suppose que la taille de la fenêtre est de 2 kilo-octets, une fois le premier acquittement reçu, on va pouvoir envoyer 2 kilo-octets de données. Par la suite, on va continuer à émettre les données à un rythme de 2 kilo-octets toutes les 5 secondes. Par rapport au schéma traditionnel, on va assister à une légère déterioration des performances. La différence se marquera uniquement au début, car, par après, les débits resteront identiques. Pour un fichier de 100 kilo-octets, le schéma traditionnel prendra 250 secondes tandis que celui proposé par Nagle en prendra 254, dans l'hypothèse où aucun datagramme n'aura été perdu, endommagé ou dupliqué sur le catenet et que le délai de transit reste fixé à 5 secondes.

4.3.3.3. *Utilisation des messages ICMP de congestion*

Le but poursuivi avec la première proposition est de réduire le risque de congestion provoquée par l'arrivée d'un grand nombre de petits paquets. Avec la seconde proposition, on va présenter un mécanisme qui permet d'assurer la reprise du fonctionnement normal du catenet lorsque les premiers signes de congestion se manifestent.

Rappelons-nous que le protocole ICMP spécifie qu'un **message ICMP de congestion**[7] doit être envoyé à l'ordinateur hôte source chaque fois qu'un datagramme est détruit par une passerelle ou quand une passerelle est à court de ressource. Le protocole ICMP est catégorique : on ne peut détruire un datagramme sans envoyer en même temps un message ICMP de congestion.

L'hypothèse de base qui est faite ici est qu'un datagramme ne sera pas détruit en situation de fonctionnement normal. De plus, les concepteurs du protocole ICMP ont choisi de ne pas attendre que la passerelle soit obligée de détruire un datagramme pour envoyer un message ICMP de congestion. En pratique, ce message ICMP sera envoyé dès qu'une passerelle voit son espace disponible de mémorisation se réduire de moitié. Cette mesure a été choisie suite à la propre expérience des concepteurs de la proposition.

---

[7] Cfr. paragraphe 3.2.4. pour la description de ce message.

Chaque fois qu'un message ICMP de congestion arrive à un ordinateur hôte, le protocole TCP ou tout autre protocole du niveau bout en bout va en être averti. La réaction qui a été choisie est de réduire la quantité de données émises sur toutes les connexions ayant comme autre extrémité l'ordinateur hôte mentionnée dans le message. Pour y parvenir, le module TCP va réduire la dimension de la fenêtre associée à l'autre extrémité de la connexion. Cela signifie que cette extrémité a réduit le nombre d'octets de données qu'il est prêt à recevoir. Cette solution présente l'avantage de ne pas freiner les émissions d'accusé de réception ou de retransmission puisqu'elle agit uniquement sur l'émission de nouvelles données. En fait, la communication va continuer mais à un débit plus faible. C'est exactement l'effet recherché.

Le module TCP reviendra à un mode de fonctionnement normal une fois qu'il aura reçu un certain nombre d'accusés de réception de l'ordinateur hôte concerné.

4.3.3.4. *Critiques de ces mécanismes*

Les deux mécanismes que nous venons de présenter ont la qualité commune d'être des solutions simples et élégantes à la prévention des situations de congestion. Ils présentent toutefois deux inconvénients majeurs :

- d'une part, ils sont spécifiques au protocole TCP. Ils ne sont pas du tout transposables au protocole UDP; nous devons toutefois reconnaître que ce dernier n'est pas générateur d'un trafic important;

- d'autre part, la solution du problème de congestion relève du protocole de niveau inter-réseaux et non des protocoles de niveaux supérieurs. Cette solution sera de toute façon difficile à implémenter au niveau du protocole IP dans la mesure où ce protocole considère les datagrammes comme des unités indépendantes. Il n'en reconnait pas les utilisateurs.

CHAPITRE 5

LES INTERFACES CONCEPTUELLES DANS LE MODELE DOD

5.1. Introduction

Lors de l'étude des protocoles IP, ICMP et TCP, nous avons principalement décrit les relations horizontales qui existent entre les modules d'un même niveau mais appartenant à des systèmes (ordinateurs hôtes ou passerelles) différents. Ces relations ne sont possibles que s'il s'établit une communication verticale entre les modules des différents niveaux d'un même système. Un module d'un système ne peut généralement communiquer qu'avec les modules de niveau immédiatement inférieur et supérieur, s'ils existent. On est donc aussi amené à décrire les interactions entre les modules qui utilisent les services offerts par un protocole et les modules qui fournissent ces services. Ces interactions définissent l'interface utilisateur. Cette interface regroupe un ensemble de primitives de service qui donnent une vue conceptuelle des relations entre les modules d'un même système. Cette vue est conceptuelle dans le sens où elle est indépendante de toute implémentation.

Dans ce chapitre, nous allons évidemment décrire les interfaces utilisateurs pour les protocoles IP et TCP. Nous avons également choisi de l'interface utilisateur du protocole Ethernet dans un double souci :

- un souci de complétude : il n'existe en effet dans l'architecture DoD que trois niveaux pour lesquels une interface utilisateur est définie; il s'agit des niveaux accès-réseau, inter-réseaux et bout en bout; pour le niveau applications, aucune interface utilisateur ne peut être définie puisqu'il n'existe aucun niveau qui lui est supérieur;

- un souci de parallélisme : dans le chapitre suivant, nous allons donner un exemple d'implémentation des interfaces utilisateurs; pour cet exemple, nous avons choisi l'interface Ethernet pour le niveau accès-réseau.

Pour spécifier les primitives de service, nous avons fixé les conventions suivantes:

- les primitives ainsi que leurs arguments seront mentionnés en français bien que dans la littérature, ils soient le plus souvent exprimés en anglais;

- la syntaxe utilisée pour décrire les primitives sera proche de celle utilisée pour les déclarations en Pascal; comme il est parfois difficile de donner un type à certains arguments, nous avons choisi de ne pas spécifier le type des arguments;

- toutefois pour les arguments qui se trouvent également comme champ dans les unités de données échangées, on indiquera entre parenthèses, leur longueur et éventuellement le caractère facultatif du paramètre;

- tout comme en Pascal, les arguments précédés du mot réservé VAR indiquent qu'il s'agit de résultats de l'exécution de la primitive.

5.2. Interface conceptuelle du protocole Ethernet

Ethernet est un réseau local de type CSMA/CD (Carrier Sense Multiple Access / Collision Detection). Il a été spécifié conjointement par Intel, Xerox et Digital. C'est sur la base de leur spécification (Ethernet 82) que nous allons décrire l'interface conceptuelle entre les niveaux accès-réseau et inter-réseaux.

Dans ces spécifications, on distingue deux interfaces : l'interface utilisateur et l'interface au système de gestion du réseau. Dans cette section, nous ne présenterons que l'interface utilisateur. Signalons quand même que le système de gestion du réseau remplit les deux fonctions suivantes :

- d'une part, le contrôle de la configuration; ceci comprend
  - l'initialisation, la suspension et la reprise des opérations de liaison de données de la station, et
  - la définition de l'adresse Ethernet et du mode d'adressage (normal, en diffusion, en diffusion restreinte) de la station;
- d'autre part, l'observation et le contrôle
  - des activités de liaison de données (collision, réception et émission) et
  - des erreurs rencontrées.

L'interface utilisateur comprend deux services de base; il s'agit de la transmission et de la réception d'un paquet, où le paquet est l'unité de transmission entre les interfaces Ethernet.

5.2.1. L'envoi d'un paquet

```
procedure    ENVOYER_PAQUET (
        destination;                    (48 bits)
        source;                         (48 bits)
        type;                           (16 bits)
        données;                        (0-1500 octets)
        VAR  résultat_de_transmission);
```

La primitive **ENVOYER_PAQUET** permet à l'utilisateur du service d'envoyer un bloc de données d'une longueur maximale de 1500 octets. Suivant que le mode d'adressage choisi par l'utilisateur est normal, en diffusion ou en diffusion restreinte, le paquet sera envoyé à une station particulière, à toutes les autres stations ou à plusieurs stations raccordées au réseau.

La primitive est synchrone dans le sens où l'utilisateur ne pourra continuer son exécution que quand l'opération d'envoi du paquet sera tout à fait terminée. La transmission est réussie si le paquet a pu être émis sans que l'interface ait détecté un nombre trop élevé de collisions. Remarquons bien que cela ne signifie pas que le paquet est arrivé correctement à destination. La transmission a pu échouer à cause d'un nombre excessif de collisions détectées, à cause de la détection d'une collision en dehors de la période prévue à cet effet ou à cause de l'état hors service du support de liaison de données.

5.2.2. La réception d'un paquet

```
procedure    RECEVOIR_PAQUET(
        VAR destination;                (48 bits)
        VAR source;                     (48 bits)
        VAR type;                       (16 bits)
        VAR données;                    (0-1500 octets)
        VAR résultat_de_réception);
```

La primitive **RECEVOIR_PAQUET** permet à l'utilisateur du service de recevoir un bloc de données d'une longueur maximale de 1500 octets adressé à la station. L'utilisateur a pu choisir de recevoir le paquet d'une station particulière, de n'importe quelle station connectée au réseau ou d'une partie des stations raccordées au réseau.

Comme la primitive précédente, cette primitive est synchrone. Elle ne sera donc terminée que si un paquet adressé à la station a effectivement été reçu. La réception est réussie si le paquet adressé à la station a été correctement reçu. La réception a pu rater parce que le paquet reçu a été endommagé lors de son passage sur le support physique ou parce que le paquet reçu ne compte pas un nombre entier d'octets ou encore parce que le système de gestion du réseau a arrêté les opérations de liaison de données.

5.3. Interface conceptuelle du protocole IP

L'interface utilisateur au module IP propose deux services de base. Il s'agit de la transmission et de la réception d'un datagramme. Signalons que le document (RFC 791) qui spécifie le protocole IP ne décrit que très sommairement cette interface. En particulier, il ne précise pas si les primitives sont synchrones ou asynchrones.

5.3.1. L'envoi d'un datagramme

```
procedure   ENVOYER_DATAGRAMME (
        destination;                    (32 bits)
        source;                         (32 bits)
        protocole;                      (8 bits)
        type_de_service;                (8 bits)
        durée_de_vie;                   (16 bits)
        données;
        longueur_totale;                (16 bits)
        identificateur;                 (16 bits)
        drapeau_DF;                     (1 bit)
        options;                        (0-40 octets)
        VAR résultat_de_transmission);
```

La primitive **ENVOYER_DATAGRAMME** permet à l'utilisateur d'envoyer un bloc *données* d'un ordinateur hôte *source* à un ordinateur hôte *destination* . Ces deux ordinateurs hôtes peuvent être connectés à des réseaux éventuellement différents mais interconnectés. La dimension maximale de *données* est fixée théoriquement à 65 535 octets. Dans la pratique, cette dimension varie d'un réseau à l'autre. Les spécifications prévoient toutefois que la taille maximale du datagramme doit au moins être égale à 576 octets. Rappelons que l'aspect priorité est pris en charge par l'argument *type_de_service* tandis que l'aspect sécurité est passé dans *options* .

Le module IP, quand il reçoit cet appel, va d'abord vérifier les arguments, ensuite préparer les datagrammes et enfin les envoyer. Si les arguments passés sont corrects et si le réseau a accepté d'envoyer le datagramme, la transmission est réussie. Par contre, elle échoue si le module IP détecte une faute dans les arguments passés ou si l'interface réseau n'accepte pas d'envoyer le datagramme.

5.3.2. La réception d'un datagramme

```
procedure     RECEVOIR_ DATAGRAMME (
        protocole;                      (8 bits)
        VAR destination;                (32 bits)
        VAR source;                     (32 bits)
        VAR type_de_service;            (8 bits)
        VAR longueur_totale;            (16 bits)
        VAR données;
        VAR options;                    (0-40 octets)
        VAR résultat_de_réception);
```

La primitive **RECEVOIR_DATAGRAMME** permet à un module associé à *protocole* de recevoir un bloc *données* d'un ordinateur hôte *source* spécifié ou non. Cet ordinateur et l'ordinateur hôte local sont connectés à des réseaux éventuellement différents qui sont alors interconnectés.

Quand un datagramme provenant du réseau arrive au module IP destination, soit il existe un appel RECEVOIR_DATAGRAMME en attente pour ce datagramme, soit il n'en existe pas. Dans le premier cas, l'appel sera satisfait et les arguments seront complètés. Dans le second cas, l'utilisateur désigné par le champ protocole du datagramme est averti de l'arrivée d'un datagramme. Si cet utilisateur n'existe pas, un message ICMP de **destination hors de portée** est envoyé à l'émetteur du datagramme et les données sont détruites. L'appel restera en suspens tant qu'aucun datagramme correctement adressé n'est arrivé. La réception ne ratera que dans le cas où une erreur est détectée dans les arguments passés.

5.4. Interface conceptuelle du protocole TCP

L'interface qui nous reste à étudier est celle qui relie les processus applications au module TCP. C'est à travers cette interface qu'un processus application pourra disposer des ressources du réseau.

Cette interface a fait l'objet de spécifications conceptuelles plus précises que celle du module IP. Elle comprend des primitives qui permettent au processus application d'établir une connexion, d'envoyer et de recevoir des données à travers la connexion établie, de fermer une connexion, de détruire une connexion ainsi que d'obtenir des informations concernant l'état de la connexion. L'interface permet également au module TCP de fournir des informations aux processus qui l'utilisent. C'est par ce mécanisme que les erreurs survenues lors de l'exécution de ces primitives sont renseignées au processus application.

5.4.1. L'ouverture d'une connexion

Le protocole TCP prévoit quatre manières d'ouvrir une connexion.

5.4.4.1. *L'ouverture d'une connexion passive avec un utilisateur éloigné non spécifié*

```
procedure   OUVRIR_CONNEXION_NON_SPECIFIEE (
        porte_source;                   (16 bits)
        délai;                          (facultatif)
        priorité;                       (facultatif)
        sécurité;                       (facultatif)
        options[1];                     (facultatif)
        VAR  nom_local_connexion);
```

La primitive **OUVRIR_CONNEXION_NON_SPECIFIEE** permet à un processus application d'ouvrir une connexion passive à des niveaux de *priorité* et de *sécurité* définis éventuellement lors de l'appel avec n'importe quel utilisateur éloigné.

Pour ouvrir une connexion, le module TCP va d'abord vérifier que le processus qui lui demande l'établissement de la connexion peut effectivement utiliser la connexion demandée. Ceci suppose que le module TCP est capable de déterminer l'identité du processus qu'il sert et de vérifier s'il peut utiliser la connexion spécifiée.

Le module TCP ou un autre élément du système d'exploitation va ensuite vérifier si les privilèges de l'utilisateur qui exécute le processus sont conformes aux niveaux de *priorité* et de *sécurité* spécifiés dans l'appel. Si ces niveaux de *priorité* et de *sécurité* ne sont pas spécifiés lors de l'appel, on utilisera des valeurs par défaut pour effectuer ces vérifications. Si le processus application peut utiliser cette connexion, un bloc TCB est créé et rempli partiellement des arguments passés lors de l'appel.

Enfin, la connexion va entrer dans un état d'attente pour recevoir une demande d'établissement de connexion provenant de n'importe quel utilisateur éloigné.

Lorsque la connexion sera complètement établie, elle portera le nom *nom_local_connexion* . Ce nom permettra à l'avenir d'identifier la connexion définie par un socket local et un socket éloigné. Si le processus a défini un *délai*, il indique par là au module TCP qu'il devra détruire la connexion si les données qu'il a reçu à livrer, n'ont pas été délivrées et acquittées en déans le délai indiqué. Si, lors de l'appel, aucun délai n'a été spécifié, le délai est fixé à 5 minutes.

---

[1] Ce champ sert à transmettre des options du protocole IP et du protocole TCP.

5.4.1.2. *L'ouverture d'une connexion passive avec un utilisateur éloigné spécifié*

```
procedure   OUVRIR_CONNEXION_SPECIFIEE (
        porte_source;                   (16 bits)
        socket_éloigné;                 (48 bits)
        délai;                          (facultatif)
        priorité;                       (facultatif)
        sécurité;                       (facultatif)
        option;                         (facultatif)
        VAR nom_local_connexion);
```

La primitive **OUVRIR_CONNEXION_SPECIFIEE** permet à un processus application d'ouvrir une connexion passive à des niveaux de *priorité* et de *sécurité* définis éventuellement lors de l'appel avec l'utilisateur éloigné spécifié par *socket_éloigné* .

Dans ce cas, la connexion ne sera complètement établie que lorsque le module TCP aura reçu une demande d'établissement de connexion provenant d'un utilisateur éloigné spécifié par *socket_éloigné* .

5.4.1.3. *L'ouverture d'une connexion active*

```
procedure   OUVRIR_CONNEXION_ACTIVE (
        porte_source;                   (16 bits)
        socket_éloigné;                 (48 bits)
        délai;                          (facultatif)
        priorité;                       (facultatif)
        sécurité;                       (facultatif)
        option;                         (facultatif)
        VAR nom_local_connexion);
```

La primitive **OUVRIR_CONNEXION_ACTIVE** permet à un processus application d'ouvrir une connexion active à des niveaux de *priorité* et de *sécurité* définis éventuellement lors de l'appel avec l'utilisateur éloigné spécifié par *socket_éloigné* .

Dans ce cas, le module TCP va commencer une procédure d'établissement de connexion en trois étapes avec l'utilisateur éloigné spécifié par *socket_éloigné* . La connexion ne sera complètement établie qu'au terme du déroulement correct de cette procédure.

5.4.1.4. *L'ouverture d'une connexion active avec l'envoi de données*

```
procedure OUVRIR_CONNEXION_ACTIVE_ENVOYER_DONNEES (
        porte_source;                   (16 bits)
        socket_éloigné;                 (48 bits)
        délai;                          (facultatif)
        priorité;                       (facultatif)
        sécurité;                       (facultatif)
        option;                         (facultatif)
        données;
        longueur_données;
        drapeau_PUSH;                   (1 bit)
        drapeau_URGENT;                 (1 bit)
        VAR nom_local_connexion);
```

La primitive **OUVRIR_CONNEXION_ACTIVE_ENVOYER_DONNEES** permet à un processus application d'ouvrir d'abord une connexion active avec l'utilisateur éloigné spécifié par *socket_éloigné* à des niveaux de *priorité* et de *sécurité* définis éventuellement lors de l'appel et ensuite, d'envoyer des *données* à l'autre extrémité à travers la connexion qui vient d'être établie. Cet aspect de la primitive est décrit au paragraphe suivant.

5.4.2. L'envoi de données à travers la connexion établie

```
procedure   ENVOYER_DONNEES (
        nom_local_connexion;
        données;
        longueur_données;
        drapeau_PUSH;                   (1 bit)
        drapeau_URGENT;                 (1 bit)
        délai;                          (facultatif)
        VAR résultat_transmission);
```

La primitive **ENVOYER_DONNEES** permet à un processus application d'envoyer des *données* à travers la connexion spécifiée par *nom_local_connexion.*

Si la connexion n'a pas pu être établie ou si le processus application n'est pas autorisé à utiliser cette connexion, une erreur est retournée. Si les données n'ont pas été envoyées et acquittées à la fin du délai fixé, la connexion sera purement et simplement détruite.

Si *drapeau_PUSH* est mis, cela provoquera l'envoi immédiat des données qui attendaient pour être envoyées à l'autre extrémité de la connexion. Le segment TCP qui contient les dernières données passées lors de l'appel aura le champ PUSH mis à 1. Si *drapeau_PUSH* n'a pas été mis, les données à envoyer pourront être combinées avec des données provenant d'autre appels pour des raisons d'efficacité.

Le processus application peut indiquer que les données à envoyer sont des données urgentes avec *drapeau_URGENT*. Les segments TCP qui transportent ces données urgentes auront le champ URG mis à 1.

Si la valeur de délai a été spécifiée, c'est cette nouvelle valeur qui devra être prise en considération pour la connexion.

La spécification de l'interface utilisateur du module TCP prévoit que la primitive ENVOYER_DONNEES peut être synchrone ou asynchrone. Dans le premier cas, le processus application ne récupérera la main qu'une fois la transmission correctement achevée ou le délai d'attente écoulé. Dans le second cas, le processus application peut continuer à travailler alors que le module TCP exécute l'opération ENVOYER_DONNEES. Un processus application pourra donc avoir à un moment donné plusieurs opérations ENVOYER_DONNEES qui s'exécutent. Elles seront traitées suivant la discipline FCFS (First Come, First Served).

5.4.3. La réception de données à travers la connexion établie

```
procedure   RECEVOIR_DONNEES (
        nom_local_connexion;
        espace_alloué_données;
        longueur_espace;
        VAR   longueur_données_reçues;
        VAR   drapeau_URGENT;        (1 bit)
        VAR   drapeau_PUSH;          (1 bit)
        VAR   résultat_réception);
```

La primitive **RECEVOIR_DONNEES** permet à un processus application d'allouer de l'espace supplémentaire pour recevoir de nouvelles données sur la connexion spécifiée par *nom_local_connexion* .

Si la connexion n'a pas pu être établie ou si le processus application n'est pas autorisé à utiliser cette connexion, une erreur est retournée.

L'exécution de la primitive ne sera complètement terminée que quand l'espace alloué aux données à recevoir a été entièrement rempli ou quand le module TCP a reçu pour cette connexion un segment TCP avec le champ PUSH mis à 1. Dans ce cas, la longueur des données reçues diffère de celle de l'espace alloué.

Si le module TCP reçoit un segment TCP contenant des données urgentes, c'est-à-dire un segment TCP dont le champ URG est mis à 1, le processus application en sera immédiatement averti par l'arrivée d'un signal envoyé par le module au processus. Ce que fera le processus à la réception de ce signal n'est pas spécifié par le protocole TCP, mais on peut supposer que le processus va chercher à traiter ces données le plus vite possible. On observera encore que les données non urgentes ne peuvent pas être délivrées au processus application dans le même espace que celui des données urgentes à moins que la séparation entre les deux types de données soit évidente pour le processus application.

Comme pour la primitive ENVOYER_DONNEES, la spécification de l'interface utilisateur du module TCP prévoit que la primitive RECEVOIR_DONNEES peut être synchrone ou asynchrone. Dans ce dernier cas, plusieurs opérations peuvent s'exécuter à un moment donné; pour permettre au processus application de distinguer les différents appels coexistants et pour tenir compte du fait que les espaces alloués ne sont pas toujours remplis, on ajoutera au code retourné, la référence à l'espace alloué ainsi que la longueur des données reçues.

5.4.4. La fermeture de la connexion

```
procedure  FERMER_CONNEXION (
        nom_local_connexion;
        VAR  résultat_fermeture),
```

La primitive **FERMER_CONNEXION** permet à un processus application de fermer la connexion spécifiée par *nom_local_connexion* .

Si la connexion n'était pas ouverte au moment de l'appel ou si le processus n'est pas autorisé à utiliser cette connexion, une erreur est retournée.

La fermeture de la connexion doit être propre dans le sens où toutes les primitives dont l'appel a été effectué avant la fermeture de la connexion et qui attendent encore d'être terminées, vont encore être exécutées avant la fermeture de la connexion.

De plus, la fermeture de la connexion ne doit pas empêcher le processus application de continuer à recevoir des données provenant de l'autre extrémité de la connexion. En fait, la primitive FERMER_CONNEXION signifie "*je n'ai plus rien à envoyer*" et non pas "*je ne veux plus rien recevoir*".

Comme une fermeture de connexion nécessite un échange d'information avec le module éloigné, la connexion peut rester dans un état de fermeture (clos**ing**) un certain temps avant d'être effectivement fermée.

5.4.5. La destruction d'une connexion

```
procedure   DETRUIRE_CONNEXION (
        nom_local_connexion;
        VAR  résultat_destruction);
```

La primitive **DETRUIRE_CONNEXION** permet à un processus de détruire une connexion établie.

Si le processus n'est pas autorisé à détruire cette connexion, une erreur est signalée.

La destruction d'une connexion va provoquer la destruction des opérations ENVOYER_DONNEES et RECEVOIR_DONNEES qui ne sont pas encore achevées, la destruction du bloc TCB et l'envoi d'un message spécial RESET qui indique au module TCP éloigné que la connexion doit être détruite.

5.4.6. La demande d'informations relatives à une connexion

```
procedure   ETAT_CONNEXION (
        nom_local_connexion;
        VAR  données_état_connexion);
```

La primitive **ETAT_CONNEXION** permet à un processus application de s'informer de l'état dans lequel se trouve une connexion.

Un processus ne peut demander que des informations d'état relatives aux connexions qu'il peut utiliser. Ces informations proviennent du bloc TCB.

Cette primitive est très dépendante de l'implémentation et elle peut être très bien laissée de côté sans pour cela entraîner des conséquences néfastes. Remarquons que certaines informations peuvent ne pas encore être disponibles ou pertinentes. Ceci dépend de l'état de la connexion au moment de l'appel ou en fonction de l'implémentation.

Ces informations sont les suivantes :

- le socket local,
- le socket éloigné,
- le nom local de la connexion,
- la fenêtre en réception,
- la fenêtre en émission,
- l'état dans lequel se trouve la connexion,
- le nombre d'espaces de données en attente d'être émis ou acquittés,
- le nombre d'espaces de données en attente d'être remplis,
- un drapeau pour indiquer que le processus a des données urgentes à traiter,
- le degré de sécurité de la connexion,

- le degré de priorité de la connexion,
- la valeur du délai.

5.4.7. Messages d'informations du module TCP au processus application

On part ici de l'hypothèse que le système d'exploitation fournit au module TCP un moyen de signaler de manière asynchrone l'arrivée d'un évènement au processus application. Quand le module TCP avertit le processus application, les informations transmises peuvent être des messages d'erreur quelle que soit la procédure, ou le résultat de l'exécution des primitives ENVOYER_DONNEES et RECEVOIR_DONNEES.

Au tableau 5.1., nous avons repris les informations transmises dans les messages auxquelles nous avons fait correspondre les primitives qui provoquent ces messages.

Informations reprises dans le message	Primitives qui ont provoqué l'envoi du message
nom local de la connexion	toutes les primitives
information spécifique du message	toutes les primitives
espace de données concerné	RECEVOIR_DONNEES et ENVOYER_DONNEES
longueur de l'espace de données	RECEVOIR_DONNEES
drapeau PUSH	RECEVOIR_DONNEES
drapeau URGENT	RECEVOIR_DONNEES

Tabl. 5.1. - Tableau récapitulatif des messages d'informations.

CHAPITRE 6

EXEMPLE D'IMPLEMENTATION DES INTERFACES DU MODELE DOD : LA VERSION 4.2BSD DE UNIX

6.1. L'implémentation de la version 4.2BSD de UNIX

Dans ce chapitre, nous allons décrire comment les interfaces de certains modules du modèle DoD ont été implémentées dans la version 4.2BSD. Ici comme au chapitre précédent, nous nous limiterons à présenter l'interface d'un seul module de chacun des niveaux accès-réseau, inter-réseaux et bout en bout. Pour les niveaux inter-réseaux et bout en bout, les modules IP et TCP s'imposent. Pour le niveau accès-réseau, nous avons choisi l'interface Ethernet pour les raisons évoquées à la section 5.1. Nous n'étudierons donc pas les interfaces des modules UDP et ICMP ainsi que leurs interactions avec les modules IP et TCP.

La principale caractéristique de cette implémentation est de permettre aux processus d'application de ne communiquer entre eux qu'au travers d'un socket. Un **socket** est défini en général comme étant un point extrémité d'une communication entre processus. Dans le cas d'une communication entre deux ordinateurs hôtes, ces processus sont des processus d'application. L'introduction de ce concept dans UNIX a entraîné la création d'un nouveau niveau entre les niveaux bout en bout et applications. Ce niveau est appelé **niveau socket**. Il a pour tâche de convertir des demandes de communication exprimées par les processus d'application en appels aux modules TCP ou UDP.

Au chapitre 2, nous avions réparti les fonctions du protocole IP en deux groupes. L'un était constitué des fonctions d'harmonisation spécifiques à chaque interface réseau tandis que l'autre regroupait toutes les fonctions du protocole IP qui étaient indépendantes des réseaux connectés. Si nous regroupons l'ensemble des fonctions d'harmonisation dans un **sous-module** qualifié **d'harmonisation** et l'ensemble des fonctions restantes dans un **sous-module commun**, nous définissons de la sorte une nouvelle interface entre le sous-module d'harmonisation et le sous-module commun. Dans l'implémentation de la version 4.2BSD du modèle DoD, cette nouvelle interface est clairement mise en évidence.

La figure 6.1. représente les interactions que nous venons de décrire entre les modules d'un ordinateur hôte.

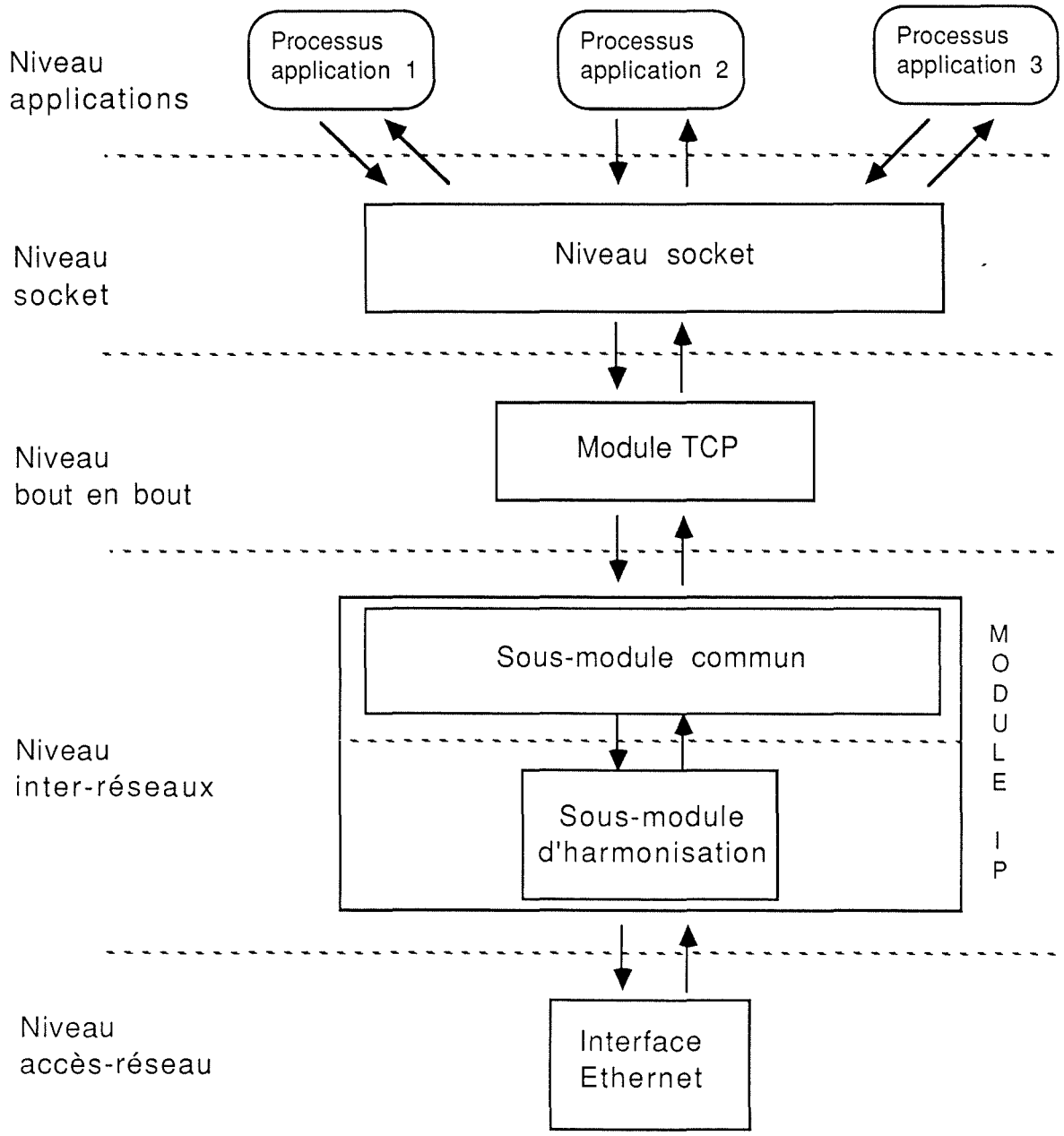


Fig. 6.1. - Interactions entre les modules à l'intérieur d'un ordinateur hôte.

La suite de ce chapitre va être organisée de la manière suivante : à la section 6.2., nous présenterons sommairement l'interface au réseau Ethernet; à la section 6.3., nous décrirons l'interface au sous-module d'harmonisation; ensuite suivront aux sections 6.4. et 6.5., les descriptions des interfaces du module IP et du module TCP; à la section 6.6., nous présenterons le niveau socket. Enfin, à la section 6.7., nous résumerons à l'aide d'un graphique les interfaces décrites dans ce chapitre.

Pour rédiger ce chapitre, nous avons utilisé, d'une part, l'article (Leffler 83) consacré à la description de l'implémentation des aspects de communication dans la version 4.2BSD de UNIX et, d'autre part, le manuel de commandes UNIX. Pour l'essentiel, ce chapitre se base sur l'analyse du code de la version 4.2BSD.

6.2. Interface au réseau Ethernet

6.2.1. Introduction

L'interface que nous avons choisie fournit un accès à un réseau Ethernet caractérisé par une technique d'accès CSMA/CD et par un débit de 10 mégabits par seconde. Cet accès se fait à travers un contrôleur 3COM qui dispose d'une mémoire propre de 32 kilo-octets directement adressables à partir de l'UNIBUS. Cette interface remplit, entre autres, les trois fonctions suivantes :

- l'implémentation du protocole CSMA/CD pour l'envoi et la réception de paquets; cette fonction est remplie pour l'essentiel par le contrôleur;
- l'échange de données et de signaux de contrôle entre la station et le contrôleur;
- la mise à la disposition des modules de niveaux supérieurs d'une interface utilisateur.

Cette interface fournit aux modules qui l'utilisent des mécanismes pour envoyer et recevoir un paquet.

6.2.2. L'envoi d'un paquet Ethernet

Le module qui veut utiliser l'interface Ethernet pour envoyer un paquet devra d'abord construire ce paquet à l'exception de la somme de contrôle. Ensuite, il placera le paquet dans la file d'attente des paquets Ethernet à transmettre. Cette file est dénommée *ifp->if_snd* . Enfin, le module appellera la fonction *ecstart* . Cette fonction traitera un à un tous les paquets en attente dans la file. Chaque paquet sera transmis à la mémoire du contrôleur où commencera la procèdure CSMA/CD pour l'envoyer.

6.2.3. La réception d'un paquet Ethernet

En permanence, il s'exécute une fonction *ec_rint* [1] qui reçoit les interruptions (*rint* signifie *receiver interrupt*) provenant du contrôleur. Quand une interruption arrive, la fonction va aller chercher le paquet reçu dans la mémoire du contrôleur. Ensuite, il sera adressé au module supérieur adéquat. Actuellement, dans la version 4.2BSD disponible à l'institut, il n'est prévu que deux modules supérieurs : il s'agit du module IP et du module ARP. Pour le module IP, le paquet sera décapsulé de son en-tête Ethernet et il sera ajouté à la file d'attente des datagrammes reçus des différentes interfaces. Cette file d'attente s'appelle *ipintrq*. Elle est implémentée de la même manière que *ifp->if_snd*. Elles seront décrites plus en détail par la suite.

6.3. Interface au sous-module d'harmonisation du réseau Ethernet

6.3.1. Introduction

Pour comprendre comment s'organisent les interactions entre le sous-module commun et le sous-module d'harmonisation à l'intérieur du module IP, nous devons d'abord décrire un certain nombre de mécanismes relatifs

- à la représentation interne des adresses,
- à l'organisation de la mémoire,
- à l'organisation de l'envoi de datagrammes et
- à l'organisation de leur réception.

Après avoir présenté ces mécanismes, nous pourrons décrire la primitive qui réalise les fonctions d'harmonisation quand le module IP a à envoyer un datagramme à travers l'interface Ethernet. Comme nous le constaterons au paragraphe 6.3.5, la réception des datagrammes est organisée de telle manière qu'elle n'exige aucune opération d'harmonisation pour l'interface Ethernet comme pour les autres interfaces.

---

[1] *rint* signifie *receiver interrupt*.

6.3.2. Représentation interne des adresses

L'idée qui a guidé la conception de toutes les adresses utilisées dans la version 4.2BSD, est de leur donner une représentation qui reste indépendante du réseau connecté ou de l'architecture choisie. Dans ce dernier cas, cette représentation est donc aussi ouverte aux adresses des architectures SNA ou XNS différentes de l'adresse transport définie dans l'architecture DoD.

Nous pouvons fournir deux justifications à un tel choix. D'une part, cela permet d'étendre les services de communication offerts par les sockets à d'autres services que ceux fournis par le modèle DoD. D'autre part, cela augmente la simplicité de la gestion des tables de routage. A la limite, on pourrait très bien n'utiliser qu'une seule table de routage même si différentes architectures coexistent sur l'ordinateur hôte.

La représentation interne qui a été adoptée pour les adresses est la suivante.

```
struct sockaddr {
        short sa_family;
        char sa_data[14];
        };
```

Toute adresse appartient au moins à une famille d'adresse. C'est la famille d'adresse qui définit le format et l'interprétation de l'adresse. Le champ *sa_family* indique à quelle famille d'adresse l'adresse appartient. Le champ *sa_data* contient la valeur actuelle de l'adresse. Sa dimension de 14 octets a été choisie à la suite d'une étude des formats d'adresse adoptés dans les différentes architectures. Pour l'architecture DoD, cette adresse comprend un numéro de porte et une adresse internet. Elle est définie par la structure suivante :

```
struct sockaddr_in {
        short sin_family;
        u_short sin_port;
        struct in_addr sin_addr;
        char sin_zero[8];
        };
```

où le champ *sin_port* est un numéro de porte et *sin_addr* a une structure *in_addr* utilisée pour définir les adresses internet. La longueur des trois derniers champs est de 14 octets. Ceci correspond bien à la longueur du champ *sa_data* de la structure *sockaddr*.

6.3.3. Organisation de la mémoire

Un des points délicats d'une implémentation d'une application réseau est celui de l'organisation et de la gestion de la mémoire. Beaucoup d'informations utilisées dans ces applications ont un caractère temporaire. Nous l'avons déjà fait remarquer en ce qui concerne la gestion des tables de routage et de conversion des adresses Ethernet. Mais cela s'étend à beaucoup d'autres aspects des applications réseau. Dans la version 4.2BSD, on a adopté un seul mécanisme pour gérer dynamiquement l'allocation d'espace mémoire. Ce mécanisme est basé sur la structure *mbuf* dont la forme est la suivante :

```
struct  mbuf {
        struct mbuf *m_next;
        u_long m_off;
        short m_len;
        short m_type;
        u_char m_dat[MLEN];
        struct mbuf *m_act;
        };
```

dans laquelle MLEN est égal à 112.

Avec le champ *m_next*, on peut relier les espaces mémoires sous la forme de listes chaînées. Ces listes peuvent être reliées entre elles grâce au champ *m_act*. Ainsi, par exemple, pour mémoriser les données des différents datagrammes, les données d'un datagramme seront chaînées avec la référence *m_next* tandis que les différents datagrammes seront reliés entre eux avec le champ *m_act*. Chaque structure *mbuf* dispose d'une petite zone *m_dat* pour stocker l'information. Le champ *m_len* indique la longueur de cette zone tandis que le champ *m_off* référence le début des données mémorisées dans *m_dat* par rapport au début de la zone *mbuf*. Le champ *m_type* définit le type des données contenues dans *m_dat*. Avec ce champ, on peut indiquer par exemple qu'il s'agit de données relatives à la table de routage ou à la table de conversion d'adresses ou à tout autre type de données. Cette structure *mbuf* occupe 128 octets dont 112 de données.

6.3.4. Organisation de l'envoi de datagrammes

Suite à la décision de routage, le module IP sait à quelle station du réseau le datagramme doit être transmis et quelle interface il doit utiliser. Le sous-module commun va accéder à cette interface au travers d'une structure de données unique *ifnet* qui la décrit. Il en existe une par interface présente sur l'ordinateur hôte. Elle a la forme suivante.

```
        struct ifnet {
                char *if_name;
                short if_unit;
                short if_mtu;
                int if_net;
                short if_flags;
                short if_timer;
                int if_host[2];
                struct sockaddr if_addr;
                union {
                        struct sockaddr ifu_broadaddr;
                        struct sockaddr ifu_dstaddr;
                        } if_ifn;
                struct ifqueue if_snd;
* routines pour initialiser, utiliser et contrôler l'interface */
                int (*if_init)();
                int (*if_output)();
                int (*if_iotcl)();
                int (*if_reset)();
                int (*if_watchdog)();
/* variables statistiques concernant le comportement de l'interface */
                int if_ipackets;
                int if_ierrors;
                int if_opackets;
                int if_oerrors;
                int if_collisions;
/* autres interfaces de l'ordinateur hôte */
                struct ifnet *if_next;
                };
```

Nous n'allons donner ici que la signification des champs les plus importants de la structure *ifnet*. Le champ *if_mtu* indique la dimension maximale du champ de données des paquets échangés sur le réseau; pour le réseau Ethernet, cette valeur est de 1500 octets. Le champ *if_net* indique le numéro du réseau auquel l'interface accède. Le champ *if_flags* décrit certaines caractéristiques de l'interface. Par exemple, ce champ indique si l'interface est branchée, si elle accède à un réseau qui supporte l'adressage en mode diffusion, etc. *if_addr* est un champ de structure *sockaddr*. Il fournit l'adresse internet associée à cette interface. *if_snd* est la file d'attente des datagrammes qui doivent être transmis par cette interface. Comme nous l'avons déjà signalé, ces datagrammes sont encapsulés dans des formats de paquet propres au réseau associé à cette interface.

Avec la fonction référencée par *if_init*, on initialise l'interface; avec celle référencée par *if_output*, on envoie à travers le réseau le ou les datagrammes contenus dans cette file d'attente. Pour l'interface Ethernet, ces fonctions portent le nom de *ecinit* et *ecoutput* [2].

[2] Cfr. paragraphe 6.3.6. pour la description de la fonction *ecoutput*.

L'utilisation d'une telle structure s'explique par le fait que le sous-module commun du module IP peut envoyer un datagramme avec l'unique appel à la fonction décrite par

```
*ifp->if_output(ifp,m0,(struct sockaddr *) dst)
```

où *m0* est le datagramme à envoyer, *dst*, la destination de ce datagramme et *ifp*, l'interface à utiliser. Au moment de l'exécution, cet appel correspondra à un appel à *ecoutput* si l'interface choisie par le routage est l'interface au réseau Ethernet. Si c'est une autre interface qui a été choisie au routage, cet appel correspondra à un appel à une autre fonction.

6.3.5. Organisation de la réception des datagrammes

Tous les datagrammes qui ont été reçus par les différents interfaces ont été placés dans une file d'attente avant d'être traités par le module IP. Cette file d'attente est dénommée *ipintrq*. Elle est déclarée de la manière suivante :

```
struct ifqueue ipintrq;
```

où *ipintrq* est organisée de la même manière que *if_snd*. Il existe toutefois entre eux deux différences. D'une part, les datagrammes mémorisés dans *ipintrq* peuvent provenir de toutes les interfaces attachées à l'ordinateur hôte alors que dans *if_snd*, on ne retrouve que des datagrammes qui vont être envoyés sur une interface particulière. D'autre part, dans *if_snd*, les datagrammes sont déjà encapsulés dans des paquets tandis que dans *ipintrq*, on a effacé l'en-tête du paquet du réseau d'où il provient. Cette dernière caractéristique de la file d'attente *ipintrq* entraîne l'absence de traitement d'harmonisation pour la réception des datagrammes.

6.3.6. L'envoi d'un datagramme à travers une interface Ethernet

Quand le module IP doit envoyer un datagramme à travers une interface Ethernet, il appelle la fonction *ecoutput* qui réalise les fonctions d'harmonisation spécifiques au réseau Ethernet[3]. Cette fonction est déclarée de la manière suivante :

```
function ecoutput(ifp, m0, dst)
        struct ifnet *ifp;
        struct mbuf *m0;
        struct sockaddr *dst;
```

[3] Cfr. point 2.3.4.2.2. pour la description de ces fonctions d'harmonisation.

où *m0* est le datagramme à envoyer, *ifp* décrit l'interface utilisée et *dst* est l'adresse internet de l'ordinateur hôte ou de la passerelle à qui le datagramme doit être envoyé. La fonction *ecoutput* va effectuer les trois opérations suivantes :

- d'abord, elle va demander au module ARP de convertir l'adresse internet *dst* en une adresse Ethernet;
- ensuite, elle va construire un paquet Ethernet dans lequel le champ *type* sera déterminé par la valeur prise par *dst->sa_family* et le champ de *données* reprendra le datagramme m0; après cela, le paquet sera placé dans la file d'attente *if_snd*;
- enfin, elle appellera la fonction *ecstart* pour transmettre au contrôleur un à un les paquets en attente.

La fonction ne terminera son exécution qu'une fois l'opération d'envoi terminée (sur un échec ou sur une réussite).

6.4. Interface au module IP

6.4.1. Introduction

La version 4.2BSD a cherché à harmoniser l'accès aux services offerts par de nombreux modules. Cette harmonisation ne s'est pas limitée aux protocoles du modèle DoD. Elle a également été appliquée au protocole PUP de l'architecture XNS.

Nous allons organiser cette section de la manière suivante : au paragraphe 6.4.2., nous présenterons comment l'accès à un module a été harmonisé en général; au paragraphe 6.4.3., nous décrirons comment on accède au module IP en particulier; au paragraphe 6.4.4., nous présenterons la fonction qui permet à un module supérieur au module IP d'envoyer un datagramme sur le catenet et au paragraphe 6.4.5., nous décrirons comment le module IP réceptionne des datagrammes provenant du catenet.

6.4.2. Harmonisarion de l'accès aux modules dans la version 4.2BSD

Pour chacun des modules associés aux protocoles IP, TCP, UDP, ICMP et RIP de l'architecture DoD et au protocole PUP de l'architecture XNS, la version 4.2BSD a défini une structure *protosw* qui décrit l'accès à ces modules.

Cette structure *protosw* a la forme suivante.

```
struct protosw {
        short pr_type;
        short pr_family;
        short pr_protocol;
        short pr_flags;
/* possibilités d'accès pour les autres modules */
        int (*pr_input)();
        int (*pr_output)();
        int (*pr_ctlinput)();
        int (*pr_ctloutput)();
/* accès pour la couche socket */
        int (*pr_usrreq)();
/* fonctions utilitaires */
        int (*pr_init)();
        int (*pr_fasttimo)();
        int (*pr_slowtimo)();
        int (*pr_drain)();
        };
```

A chaque module, on a associé un ensemble de caractéristiques propres au module. Ainsi, le champ *pr_type* indique quel service de communication le protocole associé au module fournit. Ce champ n'a de sens que si le niveau socket a accès à ce module. Il définit le type de service (datagramme pour le module UDP, circuit virtuel pour le module TCP, etc.) que le niveau socket peut obtenir de ce module. Le champ *pr_family* indique à quelle architecture de protocoles le protocole appartient. Le champ *pr_protocol* identifie le protocole associé au module. Le champ *pr_flags* regroupe un ensemble de drapeaux qui caractérisent le service offert par le protocole associé au module.

Les quatre fonctions suivantes décrivent comment les autres modules peuvent accéder à ce module. La fonction référencée par *pr_input* permet aux modules inférieurs de fournir des données destinées à ce module. La fonction référencée par *pr_output* permet à des modules supérieurs d'accéder aux services de transmission de données fournis par ce module. Les fonctions référencées par *pr_ctlinput* et par *pr_ctloutput* ont le même effet que *pr_input* et *pr_output* mais pour des informations de contrôle. *pr_input* et *pr_ctlinput* sont toujours appelés d'un niveau de protocole situé au dessous du niveau du module. *pr_output* et *pr_ctloutput*, par contre, sont appelés par le module lui-même ou par un module de niveau supérieur au module décrit.

Avec la fonction référencée par *pr_usrreq*, le niveau socket a un accès direct au module.

Les quatre fonctions restantes ne sont appelées directement que par le système d'exploitation. Avant d'utiliser une quelconque fonction d'accès au module, on doit d'abord procéder à la phase d'initialisation de ce module. Cette phase peut comprendre par exemple la création et l'initialisation de tables ou de variables. Cette initialisation est réalisée par l'appel à la fonction *pr_init*. Ensuite, la fonction référencée par *pr_fasttimo* est appelée tous les 200 millisecondes tandis

que la fonction référencée par *pr_slowtimo* est appelée tous les 500 millisecondes. Ces deux fonctions effectuent des traitements qui doivent être exécutés régulièrement. La fonction référencée par *pr_drain* permet de libérer de l'espace mémoire occupé par des données qui ne sont pas cruciales. De cette manière, le système d'exploitation peut disposer d'espace mémoire supplémentaire.

Nous avons essayé de donner à la figure 6.2. une représentation des interactions théoriques que nous venons de décrire entre un module et son environnement. Le module N est le module dont nous décrivons l'accès. Le module N+1 désigne un module de niveau supérieur au module N tandis que le module N-1 désigne un module de niveau inférieur.

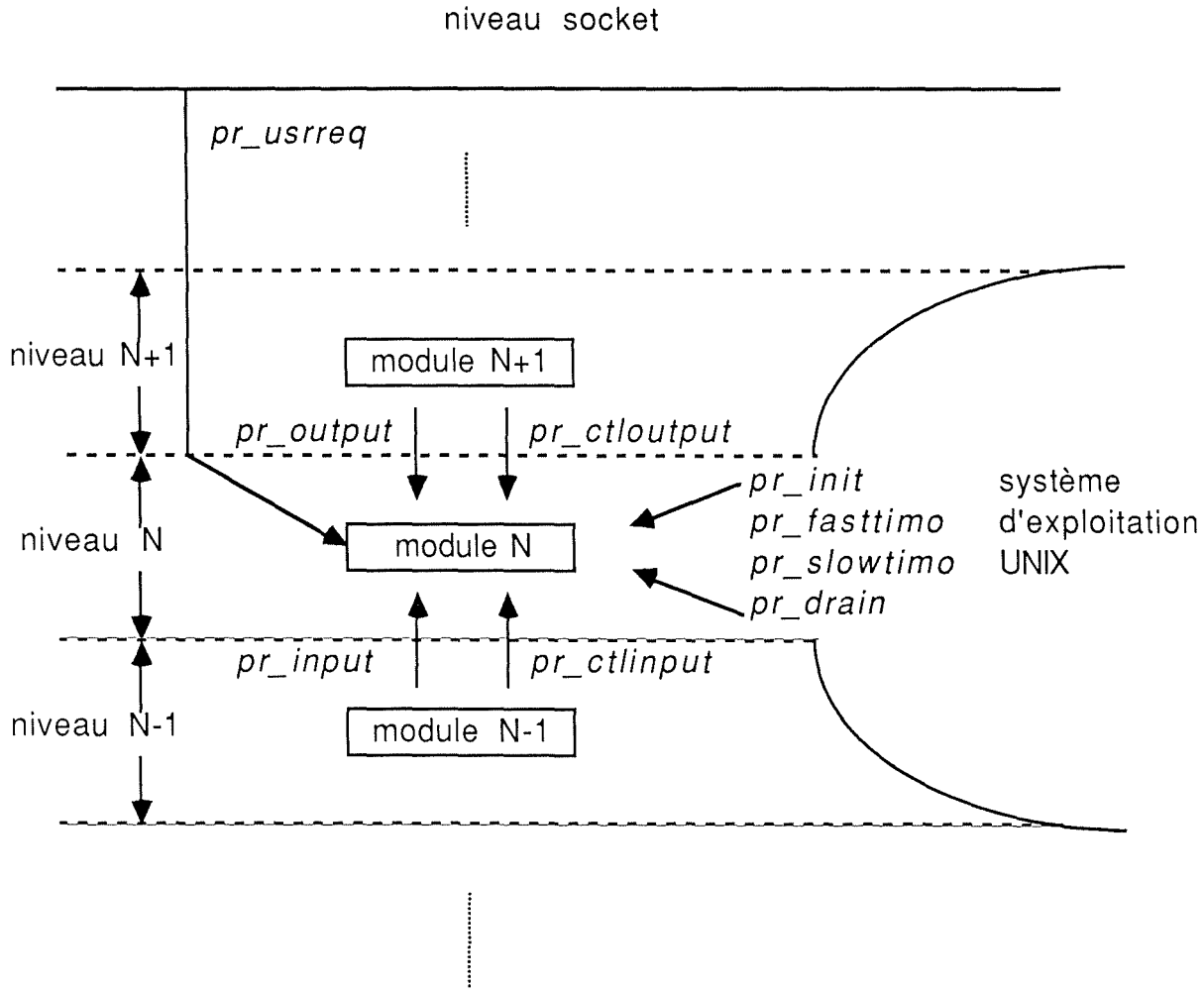


Fig. 6.2. - Interactions d'un module avec son environnement.

Cette harmonisation ne porte que sur les niveaux inter-réseaux et bout en bout, et, dès lors, beaucoup de ces interactions sont inexistantes. Nous allons directement avoir l'occasion de l'observer dans le paragraphe qui suit et qui est consacré à la présentation de la structure *protosw* du module IP.

6.4.3. Structure protosw du module IP

La valeur de la structure *protosw* pour le module IP est la suivante :

```
{
0,          PF_INET,        0,              0,
0,          ip_output,      0,              0,
0,
ip_init,    0,              ip_slowtimo,    ip_drain
}
```

Les champs *pr_type* et *pr_usrreq* sont mis à zéro pour indiquer que le module IP n'est pas accessible directement du niveau socket et donc aussi des processus d'application. Le champ *pr_family* prend la valeur PF_INET pour indiquer que le protocole IP est un protocole de l'architecture DoD. Le champ *pr_protocol* est également mis à zéro. Nous n'en connaissons pas la raison précise. Nous avons toutefois constaté que ce champ est différent de zéro uniquement pour les protocoles supérieurs à IP.

Un module de niveau supérieur au module IP peut demander au module IP d'envoyer des données en appelant la fonction *ip_output*. Il n'existe pas de fonction *ip_input* puisque, comme nous l'avons déjà indiqué, les datagrammes qui arrivent du catenet sont transmis au module IP via une file d'attente. Sur base de l'adresse destination du datagramme, le module IP va soit l'envoyer à un module de niveau supérieur au module IP, soit le renvoyer sur le catenet en appelant la fonction *ip_output*. La fonction *ip_output* peut donc être appelée du module IP lui-même ou d'un module supérieur.

Le système d'exploitation peut accéder au module IP uniquement en appelant les fonctions *ip_init*, *ip_slowtimo* et *ip_drain*.

6.4.4. Envoi d'un datagramme

Un module peut demander au module IP d'envoyer un datagramme en appelant la fonction *ip_output*. La déclaration de cette fonction est la suivante :

```
ip_output(m, opt, ro, flags)
        struct mbuf *m;
        struct mbuf *opt;
        struct route *ro;
        int flags;
```

où le paramètre *m* reprend le datagramme à envoyer sous la forme d'une liste chaînée d'espaces mémoires.

Dans cette implémentation, l'indépendance entre les niveaux inter-réseaux et bout en bout n'est pas assurée. En effet, en cas d'indépendance des niveaux, on devrait voir apparaître dans l'appel, les paramètres[4] adresse *source*, adresse *destination*, *protocole*, *type_de_service*, *durée_de_vie*, *longueur_totale*, *identificateur*, *drapeau_DF* ainsi que les *options*. En fait, ces champs d'en-tête du datagramme auront déjà été remplis par le module qui appelle la fonction *ip_output*. Ceci est contraire à une véritable séparation entre les niveaux. Ce choix d'implémentation peut se justifier par le fait que, d'une part, on ne veut pas rendre la liste des paramètres de la fonction *ip_output* trop longue et que, d'autre part, cela permet une harmonisation de l'accès aux différents modules. Le paramètre *opt* reprend la liste des options sous la forme d'une liste chaînée d'espaces mémoires.

Le paramètre *ro* est de structure route. Cette structure est définie comme suit :

```
struct route {
        struct rtentry *ro_rt;
        struct sockaddr ro_dst;
        };
```

où *ro_rt* [5] décrit le chemin pour arriver à l'ordinateur hôte ou à la passerelle dont l'adresse internet est *ro_dst*.

Si le paramètre *ro* de *ip_output* est défini, cela signifie que le module qui appelle *ip_output* impose au module IP le chemin que le datagramme a à emprunter. Sinon, c'est le module IP qui le fixe.

Le champ *flags* fournit trois indications concernant le datagramme. Une de ces indications permet au module IP de savoir si le datagramme à envoyer provient d'un module de niveau supérieur ou du module IP. Les deux autres indications sont relatives au routage et à l'adressage en mode diffusion. Nous n'avons malheureusement pas pu comprendre leur signification.

Chacun des quatre arguments de *ip_output* peut éventuellement prendre une valeur nulle.

Nous allons maintenant présenter sommairement les traitements effectués par la fonction *ip_output*.

Quand le datagramme *m* provient d'un module de niveau supérieur au module IP, les champs *identificateur*, *version* et *IHL* de l'en-tête du datagramme doivent encore être complètés. La valeur du champ *identificateur* est définie par le module IP en employant un compteur qui est incrémenté d'une unité à chaque appel. Un tel compteur est suffisant pour garantir le caractère identifiant de ce champ. Dans cette implémentation, ce champ ne peut donc pas être défini par le module de niveau supérieur au module IP comme cela avait été suggéré dans la spécification du protocole IP[6].

---

[4] Cfr. paragraphe 5.3.1. pour retrouver la liste de ces paramètres.

[5] Cfr. paragraphe 2.2.3.3. pour la présentation de la structure *rtentry* de **ro_rt*.

[6] Cfr. paragraphe 2.4.2.

Si le module de niveau supérieur n'a pas précisé le chemin que le datagramme doit suivre , alors la fonction *ip_output* devra le définir. Ceci revient à définir les variables *ifp* et *dst*. *ifp* référence la structure *ifnet* qui décrit l'interface du réseau par laquelle le datagramme doit être envoyé et *dst* référence l'adresse de l'ordinateur hôte de ce réseau à qui le datagramme devra être transmis.

Si le datagramme est trop long pour le réseau, les données du datagramme seront fragmentées en n parties dont les n-1 premières auront la dimension maximale autorisée par le réseau tout en ayant un nombre d'octets multiple de huit. La fonction *ip_output* respecte pour ce traitement les règles de fragmentation du protocole IP[7].

Après avoir calculé la somme de contrôle sur l'en-tête du datagramme éventuellement fragmenté, chaque *mf* est envoyé à l'interface *ifp* par l'appel

ifp->if_output(ifp, mf, (struct sockaddr) dst).

Si l'interface choisie est une interface Ethernet, cet appel correspond à un appel à la fonction *ecoutput* déjà décrite antérieurement.

Nous avons repris ici la liste des principales situations qui provoquent l'échec de l'envoi du datagramme.

- Dans la table de routage, on n'a pas trouvé de chemin pour arriver à *dst.*
- L'interface décrit par *ifp* pour accéder à *dst* n'est pas opérationnelle.
- *ip_output* ne dispose pas de l'espace mémoire suffisant pour fragmenter le datagramme.
- La fonction d'envoi (par exemple, *ecoutput*) signale qu'elle n'a pas réussi à envoyer le paquet qui contient le datagramme.

Dès qu'une faute est détectée, le traitement du datagramme *m* est arrêté, les espaces mémoires occupés par *m* sont libérés et un code d'erreur adéquat est renvoyé.

Si le traitement du datagramme *m* n'a provoqué aucune erreur et si l'appel à la fonction d'envoi ne retourne aucun code d'erreur , les espaces occupés par *m* sont libérés et un code de retour égal à 0 est renvoyé.

---

[7] Cfr. paragraphe 2.3.3.1 pour la description de ces règles.

6.4.5. Réception d'un datagramme

Il n'existe pas de fonction *ip_input*. Les datagrammes qui arrivent du catenet sont envoyés au module IP via une file d'attente qui est déclarée comme

struct ifqueue ipintr;

et qui regroupe tous les datagrammes qui ont été envoyés du catenet au module IP. Ces datagrammes proviennent directement des interfaces de l'ordinateur hôte. L'en-tête propre au réseau que le datagramme vient de traverser a été supprimée.

Dans le module IP, on trouve la fonction *ipintr()* qui n'a pas de paramètres. A l'appel de cette fonction, tous les datagrammes présents dans *ipintrq* sont traités jusqu'à ce que la file d'attente soit vide. Chaque fois qu'un datagramme est ajouté à la file *ipintrq*, l'interface qui reçoit ce datagramme lance l'exécution de la fonction *ipintr* par un système d'interruption. Pour l'interface Ethernet, c'est la fonction *ecread* qui provoque cette interruption.

La fonction *ipintr* opère sur chaque datagramme, les traitements relatifs aux fonctions suivantes du protocole IP :

- la détection d'erreur dans l'en-tête du datagramme;
- le traitement des options présentes dans l'en-tête du datagramme;
- le contrôle de la durée de vie du datagramme;
- éventuellement, la fonction de routage;
- éventuellement, la fonction de réassemblage.

Ces traitements s'effectuent de la manière suivante. Pour chaque datagramme *m* de *ipintrq*, la fonction *ipintr* va vérifier la somme de contrôle. Ensuite, elle va traiter les options présentes dans l'en-tête du datagramme. Si ce traitement signale une erreur, le message ICMP approprié sera envoyé à l'ordinateur hôte source du datagramme.

Sur base de l'adresse destination du datagramme, *ipintr* va vérifier si *m* est destiné à l'ordinateur hôte local. S'il n'en est pas ainsi, la durée de vie du datagramme sera réduite de 5 unités. Si la nouvelle durée de vie devient négative ou nulle, un message ICMP TEMPS DEPASSE sera envoyé à l'ordinateur hôte source du datagramme. Si la durée de vie du datagramme reste positive, *m* sera renvoyé au catenet par l'appel

ip_output(m, (struct mbuf *)0, (struct route *)0,IP_FORWARDING).

Si le datagramme est bien destiné à l'ordinateur hôte local, alors on peut essayer de réassembler le datagramme initial si *m* est un datagramme fragmenté. Cette phase de réassemblage suit les spécifications du protocole IP que nous avons décrites au paragraphe 2.3.3.2. Si le datagramme initial a pu être entièrement reconstitué ou si *m* est arrivé complet, alors il est directement transmis en-tête incluse, au module supérieur pour lequel il est destiné. Cette transmission se passe via un appel à la fonction *pr_input* de la structure *protosw* associé à ce module supérieur. Cette appel a la forme suivante :

```
(*inetsw[ip_protox[ip->ip_p]].pr_input)(m);
```

où *inetsw* est le nom du tableau qui regroupe les structures *protosw* des protocoles du modèle DoD, *ip_protox* est le nom d'une table de correspondance entre les numéros de protocole et les indices dans la table *inetsw*, de telle sorte que *ip_protox[ip->ip_p]* désigne le module dans *inetsw* à qui le datagramme est destiné. Ce module peut être le module TCP, le module UDP ou le module ICMP.

En résumé, voici les principaux motifs qui provoquent l'arrêt du traitement du datagramme :

- la somme de contrôle recalculée revèle une erreur de transmission;
- le traitement des options a détecté une erreur;
- le délai de vie du datagramme est expiré;
- le réassemblage du datagramme a échoué.

Dès qu'une faute est détectée, le traitement du datagramme est arrêté, le datagramme erroné est retiré de la file *ipintrq* et l'espace qu'il occupait est libéré.

6.5. Interface au module TCP

6.5.1 Introduction

Dans la version 4.2BSD, un processus d'application ne peut accéder directement aux services offerts par le module TCP. En particulier, il ne peut pas envoyer directement des données à travers une connexion. Il doit nécessairement utiliser le niveau socket. Ce niveau va convertir les appels système du processus d'application en appels au module TCP.

Pour envoyer des données au module TCP, le niveau socket ne dispose pas de fonction *tcp_output*. La seule manière pour le niveau socket d'accéder au module TCP est d'appeler la fonction *tcp_usrreq*. Avec cette fonction et un jeu adéquat de paramètres, le module TCP rendra tous ses services disponibles au niveau socket. A l'inverse, quand le module IP reçoit un datagramme destiné au module TCP local, le module IP le lui transmettre en appelant la fonction *tcp_input*. C'est cette fonction qui va traiter le datagramme.

Avant de présenter les fonctions d'interface du module TCP, nous devons d'abord décrire un certain nombre de structures de données. Nous commencerons, comme pour le protocole IP, par la structure *protosw* attachée au module TCP. Elle sera présentée au paragraphe 6.5.2. Ensuite nous décrirons comment le concept de bloc TCB est implémenté dans la versions 4.2BSD. Ceci fera l'objet du paragraphe 6.5.3. Enfin, au paragraphe 6.5.4., nous présenterons comment le concept de socket est représenté.

Après toutes ces descriptions, nous pourrons présenter au paragraphe 6.5.5., la manière dont le module TCP envoie et reçoit les segments et au paragraphe 6.5.6., la manière dont le niveau socket accède au module TCP.

6.5.2. Structure protosw du module TCP

La valeur de la structure *protosw* pour le module TCP est la suivante :

```
{
SOCK_STREAM,          PF_INET,          IPPROTO_TCP,
PR_CONNREQUIRED | PR_WANTRCVD,
tcp_input,            0,                tcp_ctlinput,
0,                    tcp_usrreq,       tcp_init,
tcp_fasttimo,         tcp_slowtimo,     tcp_drain
}.
```

Le protocole TCP implémente des connexions du type SOCK_STREAM. Une connexion est de type SOCK_STREAM si la communication fournie est fiable, si la transmission est duplex et si les données ne doivent pas être préalablement divisées en paquets de taille fixée avant d'être envoyés. Le champ *pr_family* prend la valeur PF_INET pour indiquer que le protocole TCP est un protocole de l'architecture définie par le DoD. IPPROTO_TCP est le numéro du protocole TCP. Sa valeur est 6. Le drapeau PR_CONNREQUIRED indique que des données ne peuvent être envoyées en utilisant le protocole que si une connexion a été préalablement établie. Le drapeau PR_WANTRCVD indique que chaque fois que le processus d'application va retirer des données reçues de l'espace mémoire lié à la connexion, les routines du niveau socket doivent en avertir le module TCP.

Les neuf champs qui suivent ont la signification qui leur a été donnée au paragraphe 6.4.2. Nous pouvons constater qu'il n'existe pas de fonctions *pr_output* et *pr_ctloutput* pour le module TCP conformément à ce que nous venons de dire à la section 6.5.1.

6.5.3. Le bloc TCB

Rappelons qu'il existe un bloc TCB dès qu'un établissement de connexion a été demandé ; il sera détruit au moment de la fermeture ou de la destruction de la connexion.

Ce bloc TCB a été décrit à l'aide des deux structures de données *inpcb* et *tcpcb*. *inpcb* décrit la connexion que est associée à ce bloc TCB tandis que *tcpcb* regroupe les variables qui sont nécessaires au fonctionnement interne du protocole TCP.

La structure inpcb est définie de la manière suivante :

```
struct inpcb {
        struct inpcb *inp_next, *inp_prev;
        struct inpcb *inp_head;
        struct in_addr inp_faddr;
        u_short inp_fport;
        struct in_addr inp_laddr;
        u_short inp_lport;
        struct socket *inp_socket;
        caddr_t inp_ppcb;
        struct route inp_route;
        };
```

Les différents blocs TCB sont reliés entre eux dans une liste circulaire en utilisant les pointeurs *inp_next* et *inp_prev*. *inp_head* référence le début de cette liste. *inp_faddr* et *inp_fport* forment l'adresse transport de l'autre extrémité de la connexion, tandis que *inp_laddr* et *inp_lport* forment l'adresse transport de ce côté-ci de la connexion. *inp_socket* référence la structure *socket* associée à un *inpcb*. La structure *socket* va être présentée dans le paragraphe qui suit. La structure *tcpcb* qui est associée à ce *inpcb* est référencée par *inp_ppcb*, où le type *caddr_t* représente un pointeur vers un caractère. Enfin, *inp_route* contient la route à suivre lorsqu'on envoie un segment TCP par un appel à la fonction *ip_output*. La structure *tcpcb* regroupe vraiment beaucoup d'informations.

```
struct tcpcb {
        struct tcpiphdr *seg_next;
        struct tcpiphdr *seg_prev;
        short t_state;
        short t_timer[TCPT_NTIMERS];
        short t_rxtshift;
        struct mbuf *t_tcpopt;
        struct mbuf *t_ipopt;
        short t_maxseg;
        char t_force;
        u_char t_flags;
        struct tcpiphdr *t_template;
        struct inpcb *t_inpcb;
/* variables nécessaires pour l'envoi de segments */
        tcp_seq snd_una;
        tcp_seq snd_nxt;
        tcp_seq snd_up;
        tcp_seq snd_w11;
        tcp_seq snd_w12;
        tcp_seq iss;
        u_short snd_wnd;
/* variables nécessaires pour la réception de segments */
        u_short rcv_wnd;
        tcp_seq rcv_nxt;
        tcp_seq rcv_up;
        tcp_seq irs;
```

```
/* variables supplémentaires non prévues dans les spécifications */
        tcp_seq rcv_adv;
        tcp_seq snd_max;
        u_short snd_cwnd;
/* variables nécessaires pour déterminer le délai de retransmission manière
dynamique */
        short t_idle;
        short t_rtt;
        short t_rtseq;
        short t_srtt;
/* variables pour les données urgentes */
        char t_oobflags;
        char t_iobc;
        };
```

Nous n'allons pas présenter ici tous les champs de cette structure. Nous allons nous limiter aux plus importants d'entre eux.

Pour les champs référencés par *seg_next* et *seg_prev*, nous devons d'abord expliquer leur structure *tcpiphdr*. Au paragraphe 6.4.4., nous avions signalé que quand un module (ici TCP) a des données (ici un segment TCP) à envoyer en utilisant le module IP, il doit remplir préalablement la plus grande partie de l'en-tête du datagramme. De la même manière, le segment TCP que le module TCP reçoit, est toujours enveloppé de son en-tête IP. Dès lors, il est logique que l'on ait défini une structure de données qui regroupe l'en-tête du datagramme internet et l'en-tête du segment TCP. Nous n'avons pas pu malheureusement trouver la signification précise de ces deux champs.

t_state décrit l'état dans lequel se trouve la connexion associée à ce bloc TCB. *t_maxseg* indique le nombre maximum d'octets que le segment peut avoir dans son champ de données. Normalement, cette valeur est au moins égale à 512.

En outre, pour que le module TCP n'ait pas à reconstituer entièrement les en-têtes IP et TCP chaque fois qu'il envoie un segment sur la connexion, il conserve dans *t_template* un exemplaire des en-têtes de IP et TCP qui contient toutes les valeurs déjà définies pour cette connexion. *t_inpcb* est un pointeur vers la structure *inpcb* associée à ce *tcpcb*.

6.5.4. La structure socket

Dans la version 4.2BSD, une structure socket est associée à chaque bloc TCB. C'est dans cette structure que le module TCP prend les segments à envoyer et qu'il place les segments qu'il reçoit. Elle a la forme suivante.

```
struct socket {
        short so_type;
        short so_options;
        short so_linger;
        short so_state;
        caddr_t so_pcb;
        struct protosw *so_proto;
/* variables utilisées pour traiter les demandes de connexion */
        struct socket *so_head;
        struct socket *so_q0;
        short so_q0len;
        short socket *so_q;
        short so_qlen;
        short so_qlimit;
/* files d'attente des données reçues et à envoyer */
        struct sockbuf so_rcv, so_snd;
/* variables restantes */
        short so_timeo;
        u_short so_error;
        short so_oobmark;
        short so_pgrp;
        };
```

Une structure *socket* est une structure de données commune au module TCP et au niveau socket. Il en existe une par connexion ouverte. Nous n'allons décrire ici que les champs qui sont directement liés à l'interface entre le module TCP et le niveau socket.

so_type définit le type de communication associée à ce socket. Pour le protocole TCP, ce champ prend la valeur SOCK_STREAM. *so_pcb* référence la structure *inpcb* associée à ce socket et *so_proto* référence la structure *protosw* qui décrit l'accès au module utilisé pour la communication. Dans le cas d'un service de type circuit virtuel, ce module est TCP. *so_rcv* est la file d'attente dans laquelle le module TCP place les segments correctement reçus dans l'ordre approprié. Cette file d'attente est accessible du processus d'application via un appel système. *so_snd* est la file d'attente dans laquelle le processus d'application peut placer les données à envoyer. Ces données seront enveloppés ensuite dans un ou plusieurs segments.

En résumé, voici représenté à la figure 6.3. les relations qui existent entre les trois structures que nous venons de décrire. Signalons encore que ces trois structures sont nécessaires pour chaque connexion ouverte ou établie par le système local.

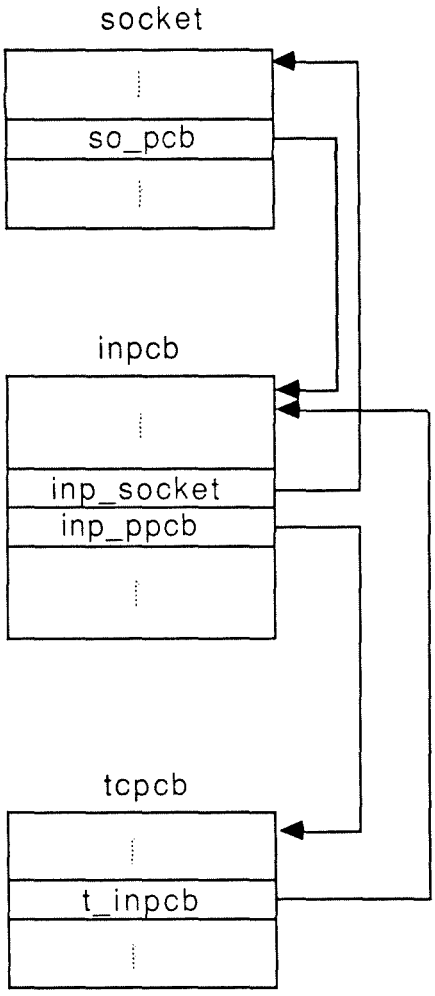


Fig. 6.3. - Relations entre les structures socket, inpcb et tcpcb.

6.5.5. Envoi et réception de segments

6.5.5.1. *Envoi de segments*

Avant de décrire comment le niveau socket accède au module TCP, nous allons d'abord présenter la fonction *tcp_output* du module TCP qui permet à un processus appartenant au module TCP d'envoyer des données des données sur une connexion.

Cette fonction est déclarée de la manière suivante :

```
tcp_output(tp)
        register struct tcpcb *tp;
```

où *tp* est une référence[8] à une structure *tcpcb* qui spécifie la connexion par laquelle les données doivent être envoyées.

Un appel à cette fonction va provoquer l'envoi sur la connexion décrite par *tp* de toutes les données qui sont mémorisées dans la file d'attente *so_snd* du socket associé à cette connexion et qu'il est possible d'envoyer. A côté des segments qui transportent des données, il existe également des segments qui ne transportent que des informations de contrôle du protocole TCP. La fonction *tcp_output* pourra être appelée pour transmettre ces deux types de segment.

Le nom attribué à cette fonction n'est peut-être pas très approprié car il serait identique au nom donné à la fonction *pr_output* de la structure *protosw* pour le module TCP si cette fonction existait. Or, comme nous l'avons déjà signalé, cette fonction n'existe justement pas pour le module TCP. En fait, quand le niveau socket demande au module TCP d'envoyer des données, il appelle la fonction *tcp_usrreq* qui à son tour appelle la fonction *tcp_output* .

Nous n'allons pas détailler dans ce paragraphe le traitement opéré par cette fonction dans la mesure où le protocole TCP n'a été présenté dans ce travail que de manière très sommaire. Nous allons donc uniquement énoncer ici les grandes étapes de ce traitement.

La fonction *tcp_output* va d'abord calculer le nombre d'octets de données qui peuvent être envoyés dans un segment. Ce calcul se base sur ce qui se trouve dans la file d'attente *so_snd*, le champ *tp->t_maxseg* et la taille des fenêtres en émission *tp->snd_wnd*.

Ensuite, on vérifie si le segment peut être envoyé. Si aucun segment ne peut être envoyé, l'exécution de la fonction *tcp_output* se termine avec un code de retour égal à 0. Si le segment peut être envoyé, la fonction va créer un espace mémoire *m* sur base du champ*tp->t_template*. Les champs de *tp->t_template* vont être recopiés et on remplit les champs qui ne sont pas encore définis de l'en-tête du segment TCP, à savoir le champ de données du segment TCP et certains champs de l'en-tête du datagramme.

Enfin la fonction *tcp_output* demande l'envoi du segment en appelant la fonction

```
ip_output(m, tp->t_ipopt, &tp->t_inpcb->inp_route, 0),
```

où *m* reprend le datagramme ayant dans son champ de données le segment TCP, *tp->t_ipopt* contient les options sélectionnées et *tp->tp_inpcb->inp_route* décrit éventuellement le chemin à suivre par le datagramme. Rappelons-nous que si ce chemin n'est pas spécifié, il sera défini par la fonction *ip_output*. S'il existe encore

---

[8] Le type *register* indique que cette référence est très utilisée.

des données à émettre, alors la fonction va essayer d'envoyer de nouveau un segment. Sinon, la fonction *tcp_output* termine son exécution en renvoyant un code de retour égale à 0.

Il n'existe que deux motifs qui provoquent une fin anormale de l'exécution de la fonction *tcp_output*. Il s'agit d'une part d'une fin anormale de l'exécution de la fonction *ip_output* et d'autre part d'un manque d'espace mémoire pour créer les segments.

6.5.5.2. *Réception de segments*

Le module IP va transmettre au module TCP le segment TCP reçu qui est toujours empaqueté dans un datagramme internet si l'adresse destination du datagramme est l'adresse internet de l'ordinateur hôte local. Pour réaliser cette transmission, le module IP va appeler la fonction *tcp_input* qui est déclarée de la manière suivante :

```
tcp_input(m0)
        struct mbuf *m0;
```

où *m0* contient le datagramme reçu avec son en-tête dans une liste chaînée d'espaces mémoires.

Le corps de cette fonction reprend la liste des actions qui doivent être entreprises pour traiter les segments qui arrivent. Les actions engagées varient en fonction de la situation dans laquelle se trouve la connexion au moment où le segment arrive. Cette fonction suit très précisement les spécifications du protocole TCP énoncées dans les pages 65 à 76 du (RFC 793).

Il nous reste deux remarques à faire à propos de la réception de segments. D'une part, la fonction *tcp_input* termine son exécution par un appel à la fonction

```
tcp_output(tp)
```

où *tp* est la structure tcpcb associée à *m0*. Ceci permettra peut-être d'envoyer de nouveaux segments alors qu'avant l'arrivée de *m0* il n'y avait aucun segment à transmettre. D'autre part, aucun code d'erreur n'est renvoyé. Le seul moyen de connaître les problèmes rencontrés se fait indirectement au travers du champ *so_error* de la structure *socket* associée à la connexion.

6.5.6. Accès du niveau socket au module TCP

Le niveau socket accède au module TCP uniquement par la fonction *tcp_usrreq*. Cette fonction est déclarée de la manière suivante.

```
tcp_usrreq(so, req, m, nam, rights)
        struct socket *so;
        int req;
        struct mbuf *m, *nam, *rights;
```

Avec l'argument *req*, le niveau socket indique quelle opération il veut que le module TCP effectue sur la connexion décrite par *so*. Nous appellerons *req* un **ordre-socket**. *m* est une liste chaînée qui contient éventuellement les données à envoyer. La signification de *nam* dépend de la valeur prise par *req*. Nous n'avons pas trouvé la signification de *rights*. *m*, *nam* et *rights* sont facultatifs.

Le paramètre *req* peut prendre 22 valeurs différentes[9]. Nous n'allons bien sûr n'en décrire que les plus importantes d'entre elles.

Avec l'ordre-socket PRU_ATTACH, la fonction *tcp_usrreq* va allouer pour *so* un bloc TCB et suffisamment d'espace pour les files d'attente de *so*. Pour chaque connexion, il est prévu un nombre fixe d'espaces mémoire. Cet ordre doit précéder tous les autres qui seront exécutés sur le socket *so*. Le bloc TCB se trouve alors dans un état fermé.

A l'inverse, *tcp_usrreq* appelé avec l'ordre-socket PRU_DETACH aura l'effet décrit dans la spécification de la fonction FERMER_CONNEXION à la différence que le niveau socket ne pourra plus non seulement envoyer des données mais aussi en recevoir. Les données qui resteraient éventuellement encore dans les files d'attente seront quand même envoyées de manière fiable.

Avec l'ordre-socket PRU_BIND, le niveau socket peut attribuer un adresse transport *nam* à la connexion définie par *so*. Dans ce cas, l'adresse internet contenue dans *nam* doit être l'adresse d'une interface présente sur l'ordinateur hôte local.

L'ordre-socket PRU_LISTEN prépare l'extrémité à recevoir des demandes de connexion. Il amène la connexion dans un état d'écoute. Avec PRU_CONNECT, la fonction *tcp_usrreq* va commencer une procèdure d'établissement de connexion en trois étapes. *nam* contiendra alors l'adresse transport de l'autre extrémité à créer. PRU_ACCEPT est utilisé après un ordre-socket PRU_LISTEN. L'adresse transport de l'autre extrémité de la connexion est transmis par *nam*.

Avec PRU_SEND, le niveau socket peut envoyer sur la connexion des données contenues dans *m*. Ces données seront placées dans la file d'attente des données à envoyer et *tcp_output* sera appelée.

---

[9] Ces 22 valeurs sont décrites dans (Leffler 83, pp. 14-16).

Si un processus d'application veut utiliser des données contenues dans la file d'attente des données reçues, ceci est signalé au module TCP par l'ordre-socket PRU_RCVD. Le seul effet de cet ordre est d'appeler la fonction *tcp_output* pour signaler à l'autre extrémité de la connexion que la fenêtre en réception est agrandie.

Avec PRU_ABORT, l'espace occupé par le bloc TCP est libéré. Le socket continue toutefois d'exister. Les données urgents peuvent être envoyées et reçues avec les ordres-socket PRU_SENDOOB et PRU_RCVOOB[10].

6.6. Le niveau socket

6.6.1. Introduction

Le niveau socket est une partie intégrante du système d'exploitation 4.2BSD dans la mesure où il fournit un grand nombre de services de communication aussi bien pour les processus qui s'exécutent à l'intérieur d'un système que pour les processus répartis sur des ordinateurs hôtes différents. Dans le premier cas, on dira que le domaine de communication est UNIX. Dans le second cas, le domaine utilisé est le domaine internet. Un **domaine de communication** est défini comme une abstraction introduite pour regrouper des propriétés communes de processus communiquant à travers des sockets.

A l'intérieur du domaine internet, les sockets ont été typés suivant les propriétés de la communication que les processus d'application peuvent attendre d'eux. Il existe cinq **types de socket**. Seul le socket STREAM nous intéresse. Un socket de ce type fournit une communication fiable et ordonnée comparable à celle offerte par le protocole TCP.

A côté de ces services de communication, le niveau socket regroupe aussi une série de routines spécifiques aux applications réseau. Ces routines permettent, entre autres, de convertir des noms d'ordinateur en adresses réseau, des noms de protocole en numéros de protocole et des noms de service en numéros de porte.

En introduction, nous avions déjà signalé qu' un processus d'application ne peut accéder aux services offerts par le module TCP que via le niveau socket. Dans cette cette section, il ne nous reste plus qu'à décrire comment les appels système provenant des processus d'application sont convertis par le niveau socket en appels au module TCP.

---

[10] Dans la version 4.2BSD, les données urgentes auront une longueur constante d'un octet.

Nous avons choisi de ne présenter que les appels système nécessaires
- pour ouvrir une connexion,
- pour envoyer des données à travers une connexion établie,
- pour recevoir des données à travers une connexion établie,
- pour fermer une connexion.

Les appels système que nous ne décrirons donc pas ici permettent d'obtenir des informations relatives à une connexion ainsi que de sélectionner des options.

6.6.2. L'ouverture dune connexion

Pour ouvrir une connexion, un processus utilisateur doit effectuer une série d'appels système qui varient suivant que la demande porte sur une connexion passive ou active.

6.6.2.1. *Ouverture d'une connexion passive*

Pour ouvrir une connexion passive, on doit d'abord créer une structure *socket* et ensuite lui attribuer une adresse transport.

Un socket est créé par l'appel système suivant :

```
s = socket(domain, type, protocol)
        int  domain, type, protocol;
```

où *domain* définit le domaine de communication choisi par le processus d'application et *type* est le type de connexion. Le paramètre *protocol* est facultatif; il définit le protocole associé à ce type de connexion. Dans le cas d'un socket de type STREAM, cet appel sera

```
socket(AF_INET, SOCK_STREAM, 0)
```

où AF_INET définit le domaine internet et SOCK_STREAM, le type de socket. L'effet de cet appel sera de créer une structure *socket* pour cette connexion; ensuite, avec l'ordre-socket PRU_ATTACH adressé au module TCP, celui-ci réservera de l'espace mémoire pour le bloc TCB et les files d'attente. Si tout se passe correctement, le processus d'application recevra un descripteur *s* qui est de type entier et qu'il devra utiliser dans chaque appel système faisant référence à ce socket particulier.

Ensuite, le processus d'application doit attribuer une adresse transport à ce socket particulier. Cela se fera avec l'appel système *bind* qui est déclaré de la manière suivante :

```
bind(s, name, namelen)
        int s;
        caddr_t name;
        int namelen;
```

où *s* est le descripteur passé par l'appel système *socket*, *name* référence l'adresse transport qui doit être associée à ce socket et *namelen* contient la longueur de cette adresse. C'est par l'ordre-socket PRU_BIND que l'adresse transport est associée au socket défini par *s*.

La connexion passera dans un état d'écoute avec l'appel système *listen* déclaré comme suit .

```
listen(s, backlog)
        int s, backlog;
```

s est le descripteur du socket associé à la connexion et *backlog* permet de limiter le nombre maximum de demandes de connexion qui peuvent arriver à ce socket. Normalement ce nombre est fixé à 5 mais il est encore possible de réduire ce nombre. Pour faire passer la connexion dans un état d'attente, le niveau socket va donner l'ordre-socket PRU_LISTEN.

Enfin, le processus d'application va entrer lui aussi dans un état inactif où il attend les demandes de connexion. Pour pouvoir traiter les demandes qui arrivent, le processus d'application lancera l'appel système *accept*.

```
ns = accept(s, name, anamelen)
        int s;
        caddr_t name;
        int *anamelen;
```

s identifie le socket mis en état d'attente; *name* est l'adresse transport de l'autre extrémité de la demande entrante et *anamelen* est la longueur de cette adresse.

L'appel système *accept* exige qu'il soit précédé d'un appel à *listen* pour la connexion concernée. Si aucune demande de connexion n'est encore arrivée, l'appel restera en attente jusqu'à ce que il en arrive une. Sinon, *accept* va prendre la première demande de connexion qui se trouve dans la file d'attente *so_q* du socket décrit par *s*. Un nouveau descripteur de socket *ns* va être renvoyé au processus d'application. Le niveau socket va transmettre au module TCP l'adresse transport de l'autre extrémité de la connexion par l'ordre-socket PRU_ACCEPT.

Si nous comparons cette interface à l'interface conceptuelle du modèle DoD, nous sommes amenés à faire les cinq observations suivantes.

- Dans la version 4.2BSD, quatre appels système sont nécessaires pour réaliser la primitive OUVRIR_CONNEXION_NON_SPECIFIEE.

- Il n'est pas possible à un processus d'application de spécifier la ou les demandes de connexion qu'il attend. Ceci signifie que la primitive OUVRIR_CONNEXION_SPECIFIEE n'est pas disponible dans la version 4.2BSD. Si certaines demandes de connexion ne sont pas désirées, le processus d'application devra tester après l'exécution de l'appel système *accept* si, oui ou non, la demande entrante lui convient. S'il refuse, il devra clôturer la connexion demandée par l'interlocuteur indésiré.

- Une ouverture passive de connexion peut toujours être rendue active si l'appel *listen* est suivi d'un appel *connect*. Dans ce cas, l'appel *connect* sera directement pris en charge.

- il n'est pas possible de transmettre une valeur de délai lors de l'établissement de la connexion. La connexion sera rompue si les données n'ont pu être livrées aprés 9 retransmissions infructueuses.

- Aucun mécanisme de *sécurité* ou de *priorité* de même qu'aucune *option* ne sont offerts dans la version 4.2BSD.

6.6.2.2. *Ouverture d'une connexion active*

Pour ouvrir une connexion active, on doit d'abord créer un socket et ensuite lui attribuer une adresse transport par un appel *bind*.

Après, on commence une procèdure d'établissement de connexion en trois étapes par un appel système *connect*.

```
connect(s, name, namelen)
        int s;
        caddr_t name;
        int namelen;
```

s est le descripteur du socket à travers lequel la connexion doit être établie; *name* est l'adresse transport de l'autre extrémité de la connexion et *namelen* est la longueur de cette adresse.

Le niveau socket va communiquer cette demande au module TCP via l'ordre-socket PRU_CONNECT. Ensuite, la fonction *connect* va attendre que la connexion soit complètement établie.

Dans la version 4.2BSD, la primitive OUVRIR_CONNEXION_ACTIVE est donc réalisée en trois appels système.

6.6.3. L'envoi de données à travers une connexion établie

Pour envoyer des données à travers une connexion, le processus d'application utilise l'appel système *send*. *send* ne peut être appelé que si la connexion a été complètement établie. Cet appel est déclaré de la manière suivante :

```
cc = send(s, buf, len, flags)
        int s;
        caddr_t buf;
        int len, flags;
```

où *s* est le descripteur associé à la connexion; *buf* référence l'endroit où se trouvent les données à envoyer et *len* est le nombre d'octets à envoyer. La valeur la plus importante que *flags* peut prendre est MSG_OOB. Ce drapeau indique que les données à envoyer sont urgentes. *cc* est le nombre d'octets effectivement envoyés.

Après cet appel, le niveau socket va placer les données dans la file d'attente des données en émission *so_snd* qui est associée à ce socket. Après quoi, elles seront envoyées sur le réseau avec l'ordre-socket PRU_SEND qui sera converti en un appel à la fonction *tcp_output*.

Le problème le plus délicat que cette fonction a à résoudre est celui de la gestion de la concurrence pour l'accès à cette file d'attente entre les différents processus attachés à ce socket. Il est en effet tout à fait possible que plusieurs processus partagent une même structure socket. Ainsi, par exemple, après l'exécution de l'appel système *fork()* [11], les deux processus (le processus parent et le processus enfant) partagent le même socket si celui-ci a été créé avant l'appel *fork*.

La fonction *send* va utiliser la technique du jeton pour organiser l'accès à la file d'attente en émission. Si plusieurs processus veulent simultanément accéder à cette file d'attente, seul celui qui aura obtenu le jeton pourra obtenir cet accès.

Si l'espace mémoire disponible dans la file d'attente est insuffisant pour placer toutes les données à envoyer, *send* va abandonner le jeton et attendre que de l'espace se libère dans la file. C'est la fonction *tcp_input* qui va avertir *send* que de l'espace a été libéré. Quand *send* aura obtenu à nouveau le jeton, elle pourra continuer à ajouter des données à la file.

Le nombre *cc* retourné au processus d'application indique le nombre d'octets qui ont été placés dans la file d'attente en émission.

---

[11] *fork* permet à un processus appelé processus parent de créer un autre processus appelé processus enfant.

Par rapport à la primitive ENVOYER_DONNEES, nous sommes amenés à faire les cinq observations suivantes :

- Par un appel à *send*, il n'y a pas moyen d'ouvrir une connexion comme cela est possible avec la primitive OUVRIR_CONNEXION_ACTIVE_ENVOYER_DONNEES.

- Un paramètre semblable à *drapeau_PUSH* n'existe pas dans la version 4.2BSD tandis que le paramètre *drapeau_URGENT* est implémenté par MSG_OOB.

- Aucun paramètre d'appel *délai* n'a été défini.

- Le processus d'application ne peut travailler tant que la fonction *send* s'exécute.

- Si les processus d'application attachés à chaque extrémité de la connexion commencent d'abord par un appel *send*, cela peut provoquer une situation d'interblocage. L'interblocage se produira par exemple si chacun des processus commence par envoyer plus de données que la somme des dimensions maximales des files d'attente en émission et en réception.

6.6.4. La réception de données à travers une connexion établie

Un processus d'application peut recevoir des données sur une connexion complètement établie en utilisant l'appel système *recv*.

```
cc = recv(s, buf, len, flags)
        int s;
        caddr_t buf ;
        int len, flags;
```

s est le descripteur du socket pour la connexion. *buf* est l'adresse de l'espace où le processus veut réceptionner les données qu'il attend. *len* est le nombre d'octets des données que le processus est prêt à recevoir. *flags* peut prendre les valeurs MSG_OOB et MSG_PEEK. Le drapeau MSG_OOB signifie que le processus veut recevoir des données urgentes. Le drapeau MSG_PEEK[12] signifie que le processus d'application veut consulter uniquement les données présentes dans la file d'attente en réception *so_rcv*. Dans le cas où le drapeau MSG_PEEK est mis, le processus ne veut pas que les données stockées dans la file d'attente soit détruite. Ainsi, un autre appel *recv* pourra encore par après être effectué sur ces données. *cc* est le nombre d'octets effectivement reçus.

---

[12] PEEK signifie coup d'oeil en anglais.

L'opération de transfert de données sera terminée dès que les *len* octets auront été transférés au processus, ou dès que toutes les données contenues dans la file d'attente auront été prélevées ou dès qu'une faute apparaît.

La fonction *recv* doit également gérer la concurrence pour accéder à la file d'attente en réception entre les différents processus. Elle utilisera aussi la technique du jeton pour organiser cet accès.

La comparaison de cette fonction avec la primitive RECEVOIR_DONNEES suscite les commentaires suivants.

- Le processus d'application n'est pas averti de l'arrivée de données urgentes. Il doit s'attendre à en recevoir pour pouvoir les recevoir.
- Comme pour la fonction *send*, le processus d'application ne peut travailler tant que la fonction *recv* s'exécute.
- Si, à chaque extrémité de la connexion, les processus d'application commencent par un appel *recv* avant un appel *send*, cela peut provoquer une situation d'interblocage.

6.6.5. La fermeture d'une connexion

Pour fermer une connexion, il existe deux appels système. Il s'agit de *shutdown* et de *close*.

L'appel système shutdown est déclaré de la manière suivante :

```
shutdown(s, how)
        int s, how;
```

où *s* est le descripteur associé à la connexion. Le résultat de l'exécution de cette fonction varie en fonction de la valeur prise par *how*.

Si *how* a la valeur 0, le processus d'application ne veut plus recevoir de nouvelles données sur la connexion. L'espace mémoire prévu pour la file d'attente en réception est détruit et chaque nouvel appel *recv* va échouer. Si *how* prend la valeur 1, le processus d'application ne veut plus envoyer de données à partir de ce socket. Cette fonction corrspond précisement à la primitive FERMER_CONNEXION de l'interface conceptuelle. Si *how* a la valeur 2, plus aucune donnée ne pourra être reçue ou envoyée.

Dans les trois cas, le socket reste intact. Il peut donc être réutilisé pour les opérations encore permises si *how* est différent de 2.

Par contre, avec l'appel *close*, le socket sera détruit. Le processus d'application ne pourra plus recevoir ou envoyer de données. Cet appel ne correspond donc pas tout à fait à l'appel FERMER_CONNEXION. Remarquons qu'il n'existe aucun appel système pour rompre une connexion.

6.7. Résumé

Nous avons repris à la figure 6.4. l'ensemble des interfaces qui existent entre les niveaux que nous venons de distinguer dans ce chapitre.

Pour expliquer cette figure, nous posons les deux hypothèses suivantes : d'une part, il existe une connexion établie entre les processus d'application A et B qui s'exécutent sur deux ordinateurs hôtes distincts; d'autre part, ces deux ordinateurs sont connectés à un même réseau Ethernet.

Pour envoyer des données à B, le processus A appelle la fonction *send*. Le niveau socket va placer ces données dans la file d'attente *so_snd* associée à ce socket et donner l'ordre au module TCP d'envoyer ces données. Ceci se fera par un ou plusieurs appels à la fonction *tcp_usrreq* avec l'ordre-socket PRU_SEND. Le module TCP va envelopper ces données dans une en-tête TCP complète et une en-tête IP incomplète. Il va ensuite transmettre le tout au module IP par un ou plusieurs appels à la fonction *ip_output*.

Le module IP va d'abord compléter l'en-tête IP et ensuite envoyer le datagramme en appelant **ifp->if_output*. Cet appel correspond en fait à un appel *ecoutput* si l'interface utilisée est l'interface Ethernet. Cette fonction *ecoutput* va encapsuler chaque datagramme dans un paquet Ethernet et le placer dans une file d'attente *ifp->if_snd*. Enfin, *ecoutput* appelera la fonction *ecstart* pour transmettre le paquet sur le réseau.

Si le paquet Ethernet arrive à l'ordinateur hôte destination et s'il existe un appel système *recv* en suspens, ce paquet va subir les traitements qui suivent.

La fonction *ec_rint* va recevoir le pâquet , lui retirer son en-tête Ethernet et le placer dans la file d'attente *ipintrq* des datagrammes reçus des interfaces présentes sur l'ordinateur hôte. La fonction *ipintr* va éventuellement essayer de reconstituer le datagramme initial et ensuite le transmettre au module supérieur par un appel

*(*inetsw[ip–protox[ip->ip–p]].pr_input)(m).*

Pour le module TCP, ceci correspond à un appel à la fonction *tcp_input* . L'effet de cette fonction sera de placer les données reçues dans la file d'attente en réception *so_rcv* associée à ce socket. Le processus d'application pourra la consulter ou y prélever des données par un appel système *recv*.

Rappelons encore que les files d'attente *so_snd* et *so_rcv* sont communes au niveau socket et au module TCP, même si, sur la figure 6.4., ils n'apparaissent qu'au niveau socket.

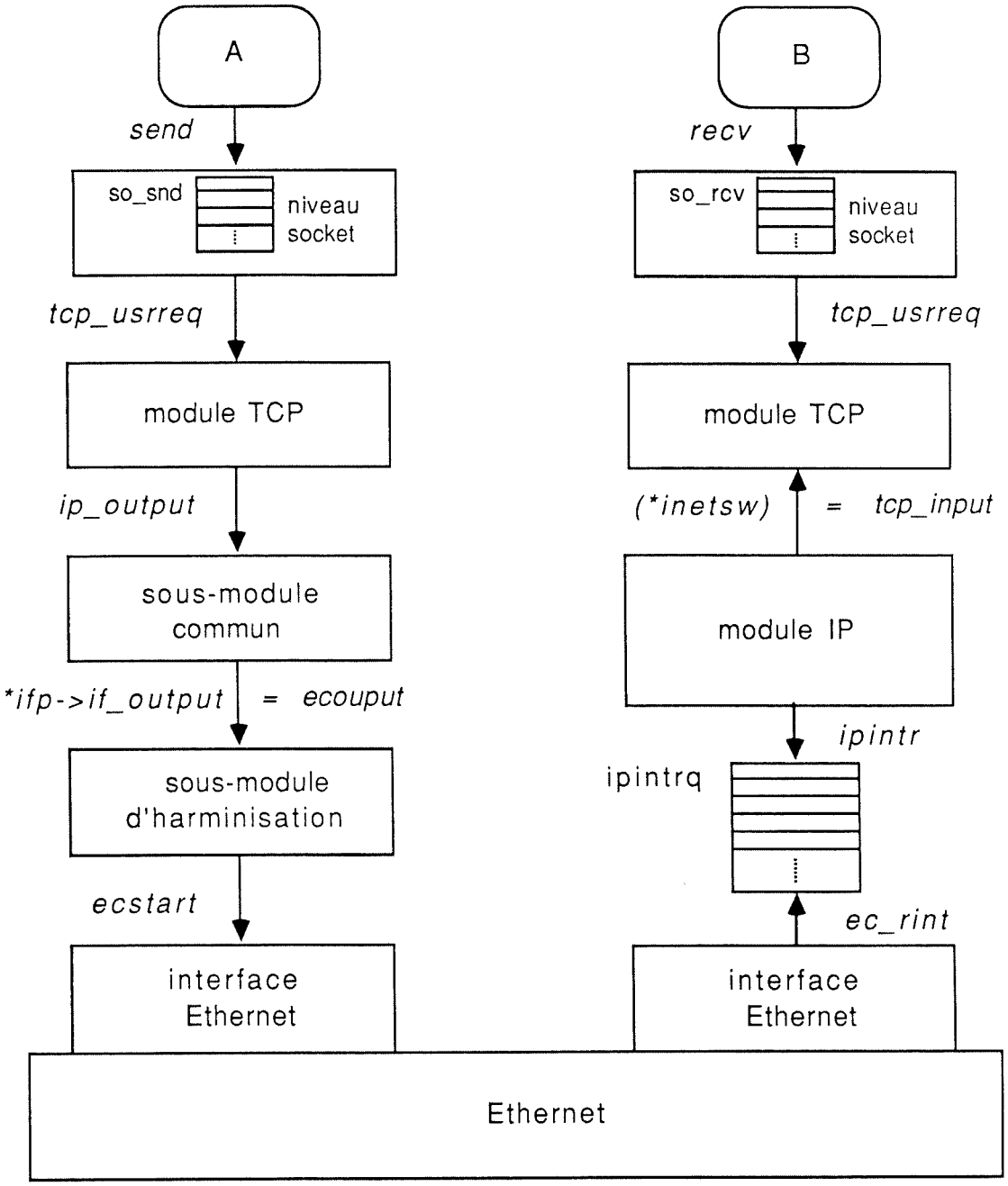


Fig. 6.4. - Exemple d'interfaces dans la version 4.2BSD.

CONCLUSION

Nous allons articuler cette conclusion autour des trois objectifs que nous nous sommes assignés en introduction. Nous les rappelons brièvement ici; il s'agit de l'étude précise

- des fonctionnalités des niveaux définis dans le modèle DoD,
- des interactions existant entre les différents modules d'un ordinateur hôte,
- des solutions alternatives proposées dans d'autres modèles.

1. Fonctionnalités des niveaux du modèle DoD

Au niveau accès-réseau, le Département de la Défense a opté pour une large utilisation des standards commerciaux existant. Ces standards ne devront toutefois pas mettre en danger le caractère opérationnel et sûr du catenet.

Au niveau inter-réseaux, nous avons distingué les fonctions communes du protocole IP de ses fonctions d'harmonisation. Les premières sont relatives à l'adressage des datagrammes, à leur routage, à leur fragmentation, au contrôle de leur durée de vie, au contrôle de leur intégrité, au choix de la qualité du service demandé par l'utilisateur du service IP, au traitement des options et au transport de données. Les secondes ont trait aux problèmes de la conversion des adresses internet en adresses sous-réseau, au choix de la taille des datagrammes à envoyer et au problème de la conversion des services offerts par les réseaux vers ceux demandés par le protocole IP.

Dans le protocole IP, il n'est prévu aucun mécanisme qui renseigne les modules IP de l'état du catenet pas plus que du résultat du traitement des datagrammes. Pour pallier à ces deux inconvénients, le modèle DoD a défini le protocole ICMP qui fournit à l'ordinateur hôte ce genre de renseignements. Les spécifications de ce protocole ne prévoient toutefois pas comment ces informations doivent être traitées par l'ordinateur hôte.

Au niveau bout en bout, le DoD a spécifié deux protocoles. Il s'agit des protocoles TCP et UDP. Dans ce travail, nous n'avons fait qu'aborder le protocole TCP. Ce protocole remplit deux fonctions importantes: il s'agit du transport fiable de données *entre processus* d'application et du contrôle de flux. A ce propos, nous avons constaté que l'architecture basée sur le protocole IP au niveau inter-réseaux et sur le protocole TCP au niveau bout en bout était la cause principale des problèmes de congestion observés sur le catenet. Pour résoudre ce problème, les solutions proposées procèdent au coup par coup, c'est-à-dire que pour chaque problème particulier, on a apporté une solution spécifique. Aucune solution générale n'est actuellement possible dans la mesure où le mode d'interconnexion utilisé est sans mémoire. Le problème posé est délicat à résoudre; il est toujours l'objet de nombreuses recherches.

Nous n'avons pas abordé, dans ce travail, les fonctions liées au niveau applications.

2. Interactions existant entre les différents protocoles

Dans ce travail, nous avons eu l'occasion de distinguer trois types d'interactions entre les protocoles à l'intérieur d'un ordinateur hôte ou d'une passerelle; il s'agit des interactions entre

- les modules associés à des protocoles du modèle DoD,
- les protocoles responsables de la gestion du catenet,
- les protocoles associés à différentes architectures de protocoles.

Le chapitre 6 a été exclusivement consacré aux interactions existant à l'intérieur d'un système entre les différents modules du modèle DoD. De cette étude, il se dégage que le critère d'indépendance entre les niveaux est souvent passé au second plan non seulement au profit de l'harmonisation de l'accès aux différents modules mais aussi au profit d'une plus grande efficacité. Ce choix se justifie dans la version 4.2BSD du modèle DoD par le fait que le concept de socket, responsable dans cette version de la communication inter-processus, ne veut pas se limiter aux deux types de communication bout en bout définie par le modèle DoD. A côté de ces deux services, la version 4.2BSD veut clairement rester ouverte à d'autres types de communication tels que celui offert par le protocole PUP de l'architecture XNS.

Pour ce qui est des protocoles responsables de la gestion du catenet, nous avons décrit avec rigueur le protocole RIP qui, d'une part, est responsable de la mise à jour des tables de routage présentes dans les passerelles et qui, d'autre part, fournit les informations de routage nécessaires aux modules IP pour leur fonction de routage. Nous avons également eu l'occasion de remarquer dans ce travail que la fonction de gestion des réseaux est l'objet à ce jour de nombreuses recherches surtout en ce qui concerne les problèmes de routage tant à l'intérieur des systèmes autonomes qu'entre les systèmes autonomes. et en ce qui concerne les problèmes relatifs au fonctionnement et à la maintenance du catenet.

Enfin, en ce qui concerne l'interopérabilité entre les architectures de protocoles, nous avons mentionné deux techniques: la plus ancienne s'appelle l'encapsulation mutuelle. Elle n'est possible que si d'une part, au niveau de la *couche réseau*, il n'existe qu'un seul protocole inter-réseaux et que d'autre part, ce protocole utilise la technique de l'encapsulation. La technique la plus récente s'appelle convertisseurs de protocoles. Ils sont principalement conçus comme étape intermédiaire vers l'utilisation universelle des normes ISO/CCITT.

3. Solutions présentes dans d'autres architectures

C'est surtout dans les chapitres 2 et 3 que nous avons eu l'occasion d'aborder des solutions alternatives à celles définies dans le modèle DoD. La constation la plus importante est, sans conteste, que les protocoles IP et ISO 8473 sont quasi identiques. La seule différence réelle se situe au niveau de la fonction d'adressage. Pour les autres fonctions associées à ces protocoles de niveau inter-réseaux, on peut constater une assez large unanimité entre les architectures.

Rappelons encore que les protocoles attachés à la gestion des réseaux font l'objet actuellement de nombreuses recherches tant dans le cadre du DoD que dans celui de l'ISO et du CCITT. Si nous avions un regret à formuler au terme de ce travail, c'est de n'avoir pas pu y consacrer plus d'importance.

4. Evaluation personnelle

Nous pouvons toutefois affirmer que les objectifs tels qu'ils ont été définis en introduction ont été atteints. Les principales difficultés que nous avons rencontrés lors de la rédaction de ce travail se situent au niveau de la précision et de la clarté des descriptions des protocoles IP, ICMP et TCP ainsi qu'au niveau de la mise en correspondance entre les traitements réalisés dans les modules de la version 4.2BSD et les fonctions remplies par les modules associés à ces protocoles.

ANNEXE A
ELEMENTS DE L'ARCHITECTURE OSI

Dans cette annexe, nous avons voulu décrire le modèle ISO dans ses aspects les plus théoriques. Nous y retrouverons donc une liste de définitions de concepts qui servent de point de départ à la spécification des normes OSI.

Le modèle OSI traite des problèmes que pose la communication entre systèmes. Un ***système*** *est un ensemble d'un ou plusieurs ordinateurs avec leurs propres logiciels, périphériques, terminaux, opérateurs humains, processus physiques, moyens de transfert de l'information, etc. Ceci forme un ensemble autonome capable d'effectuer des tâches de traitement et de transmission d'information. Un* ***processus d'application*** *est un élément du système qui traite l'information pour une application particulière. Ces processus d'application peuvent être manuels* (par exemple, la personne à un terminal Mister Cash), *informatisés* (par exemple, un programme qui demande l'accès à une banque de données éloignée) *ou physiques* (par exemple, un programme de contrôle de processus de fabrication qui s'exécute sur l'ordinateur spécialisé attaché à un équipement industriel et relié au système de contrôle de l'entreprise) (ISO 7498, pp. 4-5).

La division hiérarchique est la technique qui permet de décomposer logiquement un système individuel en une succession de sous-systèmes. Un ***sous-système*** *est donc un élément de la division hiérarchique d'un système qui ne peut interagir directement qu'avec des sous-systèmes de rang immédiatement inférieur ou immédiatement supérieur à ce système. Une* ***couche*** *est une subdivision de l'architecture OSI composée des sous-systèmes de même rang de tous les systèmes interconnectés. Chaque sous-système est constitué d'une ou plusieurs entités. Une* ***entité*** *est un élément actif à l'intérieur d'un sous-système. Une couche est composée dès lors de plusieurs entités répartis à travers les systèmes ouverts interconnectés. On appelle* ***entités-appairées****, les entités situés à l'intérieur de la même couche* (ISO 7498, pp. 9-10). La figure A.1. illustre les relations existant entre les notions de systèmes et de couches dans l'environnement OSI.

La convention de notation qui est utilisée dans ISO 7498 permet d'identifier une couche particulière par **(N)-couche**, la couche immédiatement supérieure par (N+1)-couche, la couche immédiatement inférieure par (N-1)-couche . La même notation est utilisée pour d'autres concepts qui sont associés à la notion de couche dans le modèle (ISO 7498, p. 4). Il s'agit, en autres, des concepts de service, de point d'accès au service, d'adresse, de protocole, de fonction, de connexion et d'entités correspondantes. *Les* ***(N)-services*** *sont les services fournis par la (N)-couche qui s'appuie sur les couches inférieures. Ce service est fourni aux (N+1)-entités grâce au travail collectif des (N)-entités réparties entre les systèmes interconnectés. Le* ***(N)-point-d'accès-service*** *est le point où les (N)-services sont fournis par une (N)-entité à une (N+1)-entité. Le* ***(N)-protocole*** *est l'ensemble des règles et des formats déterminant la communication entre (N)-entités dans l*

'exécution de (N)-fonctions. Une ***(N)-fonction*** *est une partie de l'activité des (N)-entités. Le contrôle de flux et le séquencement sont des exemples de (N)-fonctions. Une* ***(N)-connexion*** *est une association établie par la (N)-couche entre deux (N+1)-entités ou plus pour l'échange de données. Les* ***(N)-entités correspondantes*** *sont les (N)-entités possédant une (N-1)-connexion entre elles. Un* ***(N)-relais*** *est une (N)-fonction par laquelle une (N)-entité transmet des données reçues d'une (N)-entité correspondante à une autre (N)-entité correspondante* (ISO 7498, pp. 10-12).

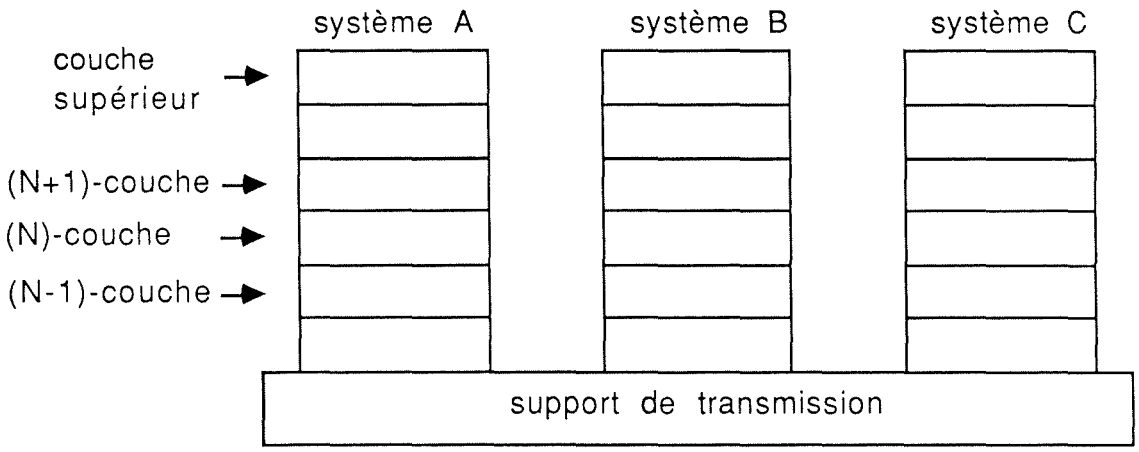


Fig. A.1. - Les concepts de système et de couche dans l'environnement OSI.

La figure A.2. illustre les relations qui existent entre les notions d'entités, de points d'accès-service (SAP's), de services, de protocoles et de couches quand plusieurs systèmes sont confondus (source : Zimmermann 83, p. 1336).

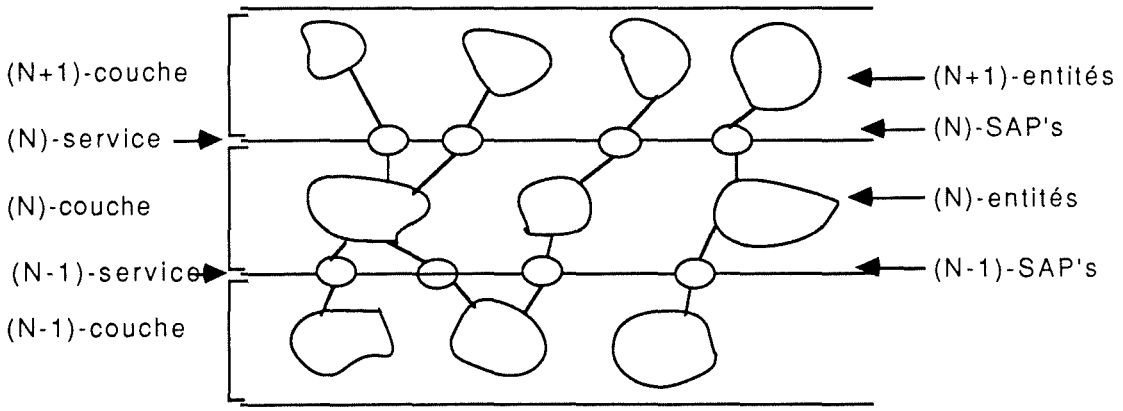


Fig. A.2. - Relations entre les concepts d'entités, de SAP's, de services, de couches et de protocoles entre plusieurs systèmes.

Les objets à l'intérieur d'une couche ou à la limite de couches adjacentes doivent être identifiables de manière unique. Ainsi, chaque (N)-entité est identifiée par un ***titre global*** *unique et permanent. Une* ***(N)-adresse*** *est un nom non-ambigu dans l'environnement OSI qui est utilisé pour identifier un ensemble de (N)-SAP's qui se situent toutes à la limite entre un (N)-sous-système et un (N+1)-sous-système dans le même système ouvert* (OSI/DIS 7498/ADD 3, p. 2). *Quand une (N+1)-entité établit une (N)-connexion avec une autre (N+1)-entité, chaque (N+1)-entité reçoit en* ***(N)-identifiant d'extrémité de connexion (CEPI)*** *de son (N)-entité. Un (N)-CEPI est un identifiant local déterminé au moment de l'établissement de la connexion* (ISO/DIS 7498/ADD3, p. 8). Les figures A.3. et A.4. représentent les relations entre ces identifiants (source : Zimmermann 83, p. 1337).

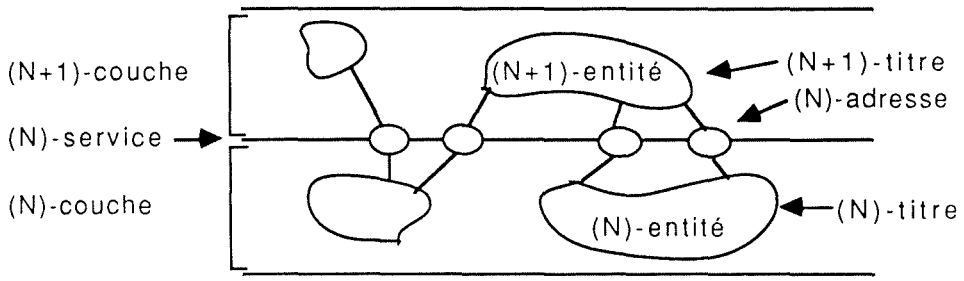


Fig. A.3.-Titres et adresses.

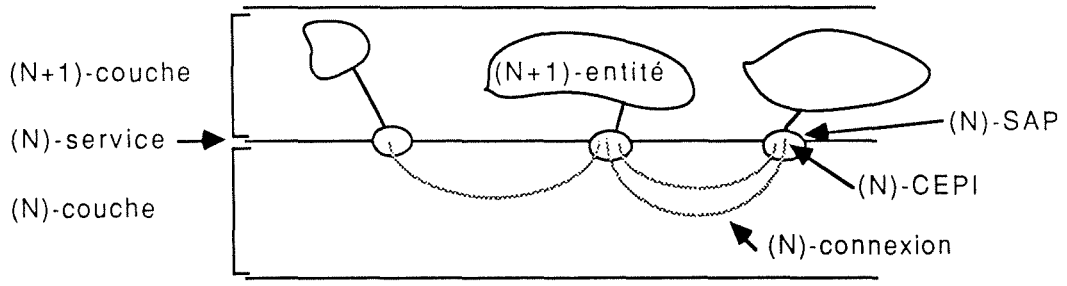


Fig. A.4. - Connexions et CEPI's.

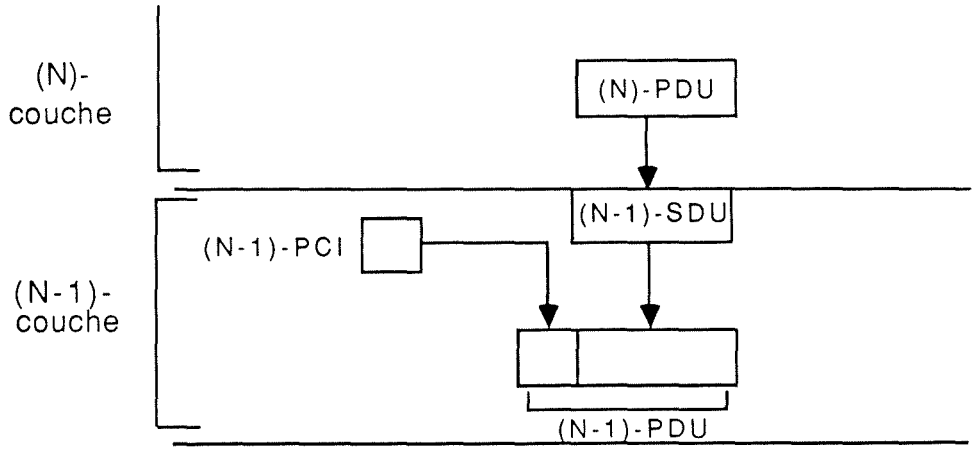
En ce qui concerne les échanges d'information, différents types d'unités de données sont prévus. Le ***(N)-information de contrôle-protocole*** *est l'information échangée entre (N)-entités, utilisant une (N)-connexion pour se coordonner. Les* ***(N)-données-utilisateur*** *sont les données transférées entre deux (N)-entités pour le compte des (N+1)-entités auxquelles elles fournissent les services demandés. Une* ***(N)-unité de données-protocole*** *est une unité de données spécifiée dans un (N)-protocole et comportant une (N)-information de contrôle-protocole et éventuellement des (N)-données-utilisateur. La* ***(N)-information de contrôle-interface*** *est l'information échangée entre une (N+1)-entité et une (N)-entité pour se coordonner. Une* ***(N)-donnée-interface*** *est l'information transférée depuis une (N+1)-entité vers une (N)-entité, et destinée à une (N+1)-entité correspondante à travers une connexion, ou inversément, il s'agit alors de l'information transférée depuis une (N)-entité à une (N+1)-entité, après avoir été reçue de la (N+1)-entité correspondante à travers la (N)-connexion. Une* ***(N)-unité de données-interface*** *est l'unité d'information transférée à travers un point d'accès service entre une (N+1)-entité et une (N)-entité lors d'une interaction. Chaque (N)-unité de données-*

interface contient une (N)-information de contrôle-interface et éventuellement tout ou partie d'une (N)-unité de données-service. Une ***(N)-unité de données-service*** *est la portion de (N)-données-interface dont l'identité est conservée entre les extrémités d'une (N)-connexion.*

Les relations entre les différentes unités de données et les problèmes de passage de ces unités entre couches adjacentes sont illustrées aux figures A.5. et A.6. (source : Zimmermann 83, p. 1337).

	contrôle	données	combinées
entités appairées (N)-(N)	(N)-information de contrôle protocole	(N)-données utilisateurs	(N)-unités de données protocole
couches adjacentes (N)-(N-1)	(N-1)-information de contrôle interface	(N-1)-données interface	(N-1)-unités de données interface

Fig. A.5. - Types d'informations circulant entre entités appairée et entités de couches adajacentes.



PDU = unité de données-protocole
SDU = unité de données-service
PCI = information de contrôle-protocole

Fig. A.6. - Correspondance entre unités de données appartenant à des couches adjacentes.

*Pour en terminer avec cette terminologie OSI, il nous reste à définir les concepts d'utilisateur et de fournisseur de service. L'**utilisateur des (N)-services** est l'entité qui, dans un système ouvert donné, utilise un service par l'intermédiaire de (N)-points d'accès au service. Le **fournisseur des (N)-services** est la représentation abstraite de la totalité des entités qui fournissent un service aux utilisateurs de services homogènes* (ISO/DIS 8509, pp. 3-4). La figure A.7. illustre les relations existant entre utilisateurs de services, fournisseurs de services et points d'accès au service (source : Pujolle 85, T1, p. 147).

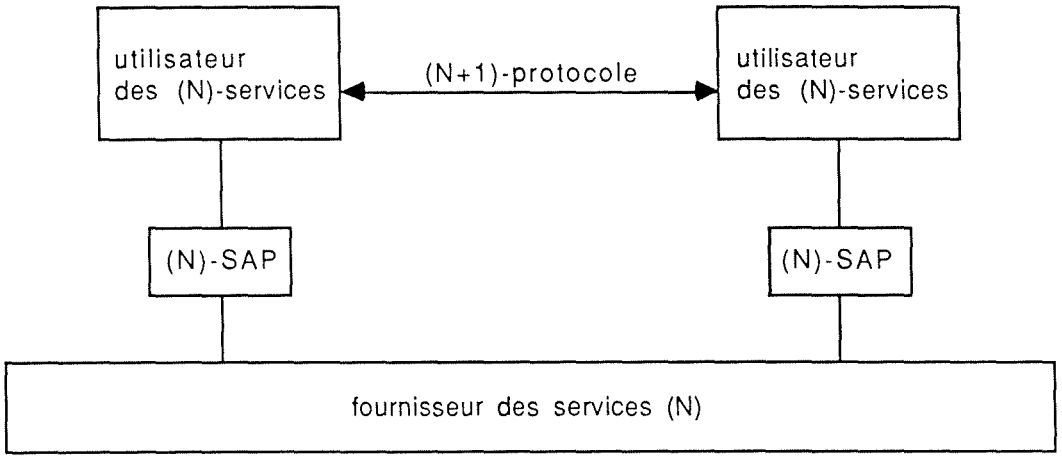


Fig. A.7. - Relation entre utilisateurs de services, fournisseurs de service et SAP's.

ANNEXE B

PRINCIPES UTILISES POUR DETERMINER LES SEPT COUCHES DU MODELE DE REFERENCE OSI

Les principes qui suivent ont été utilisés pour déterminer les sept couches du Modèle de Référence et sont présentés pour guider les décisions futures dans le développement des normes OSI.

Principe 1 : *ne pas créer trop de couches de manière à rendre la conception du système ou la description et l'intégration des couches plus difficiles que nécessaire;*

Principe 2 : *créer une limite au point où la description des services peut être limitée et où le nombre d'interactions à travers cette limite est minime;*

Principe 3 : *définir les couches de manière à manipuler les fonctions manifestement différentes dans les traitements ou la technologie impliquée;*

Principe 4 : *rassembler les fonctions similaires dans une même couche;*

Principe 5 : *choisir les limites au point où l'expérience passée a démontré que c'était source de succès;*

Principe 6 : *rassembler dans une couche des fonctions facilement localisables de telle sorte que la couche puisse être entièrement reconçue et ses protocoles profondément modifiés pour pouvoir tirer parti des progrès technologiques apparus sans pour autant changer les services demandés ou fournis aux couches adjacentes;*

Principe 7 : *créer une limite là où on peut avoir à l'avenir une interface standard;*

Principe 8 : *créer une limite là où différents niveaux d'abstraction sont nécessaires dans le traitement des données;*

Principe 9 : *permettre que des changements soient apportés dans les protocoles ou les fonctions d'une couche sans affecter les autres couches;*

Principe 10 : *attacher à chaque couche des limites uniquement avec la couche immédiatement supérieure et immédiatement inférieure.*

Des principes similaires sont appliqués à la division en sous-couches :

Principe 11 : *ne créer de nouvelles sous-couches à l'intérieur d'une couche que si des services de communication distincts sont nécessaires;*

Principe 12 : *créer, si nécessaire, deux ou plusieurs sous-couches avec une fonctionnalité commune minimale pour permettre les opérations d'interfaçage avec les couches adjacentes;*

Principe 13 : *permettre le contournement de certaines sous-couches* (ISO 7498, pp. 39-40).

ANNEXE C

EXPLICATION SOMMAIRE DE LA MANIERE DONT LES SEPT COUCHES ONT ETE CHOISIES DANS LE MODELE OSI

Cette annexe fournit une explication sommaire de la manière dont les sept couches ont été choisies.

1. *Il est essentiel que l'architecture permette que différents supports de transmission soient utilisés avec des procédures de contrôle différentes (V.24, V.25, X.21, etc.). L'application des principes 3, 5 et 8 conduit à l'identification de la couche physique comme couche la plus basse de l'architecture.*

2. *Certains supports physiques de communication (par ex., la ligne téléphonique) nécessitent des techniques spécifiques pour être utilisés lors de la transmission de données entre systèmes. Ces techniques spécifiques sont utilisées dans les procédures de contrôle de liaison de données. Elles ont été étudiées et standardisées depuis quelques années. Il existe également de nouveaux supports physiques de communication (par ex., la fibre optique) qui exigent de nouvelles procédures de liaison de données. L'application des principes 3, 5 et 8 conduit à l'identification de la couche liaison de données au dessus de la couche physique.*

3. *Dans l'architecture des systèmes ouverts, certains systèmes sont les destinataires finaux des données tandis que d'autres interviennent seulement comme noeuds intermédiaires qui transmettent les données à un autre système. L'application des principes 3, 5 et 7 conduit à l'identification de la couche réseau au dessus de la couche liaison de données. Les protocoles orientés réseau vont être groupés dans cette couche. Dès lors, la couche réseau va fournir une connexion réseau entre deux entités transport, en incluant le cas où des noeuds intermédiaires sont impliqués. Les deux fonctions les plus importantes associées à cette couche sont les fonctions d'adressage et de routage à l'intérieur d'un réseau ainsi qu'entre les réseaux.*

La figure C.1. représente une communication impliquant des systèmes de relais (source : ISO 7498, p. 39).

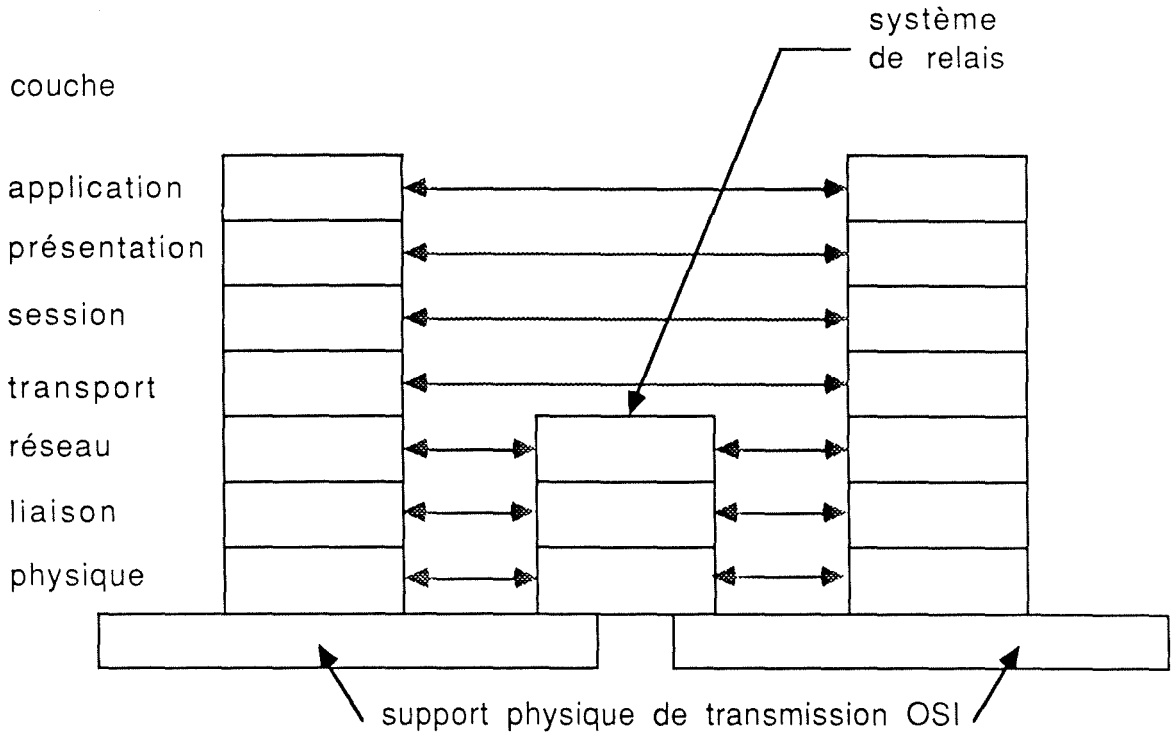


Fig. C.1. - Communication impliquant un système de relais.

4. *Le contrôle du transport des données entre un système extrémité source et un système extrémité destination est la dernière fonction nécessaire pour fournir la totalité du service transport. Donc la couche la plus élevée de la partie service transport de l'architecture est la couche transport qui vient au dessus de la couche réseau. Cette couche transport décharge les entités de la couche supérieure du souci du transport des données entre elles.*

5. *Il y a un besoin d'organiser et de synchroniser les dialogues, et de gérer les échanges de données. L'application des principes 3 et 4 conduit à l'identification de la couche session au-dessus de la couche transport.*

6. *Le reste des fonctions sont d'intérêt général et liées à la réprésentation et manipulation de données structurées au bénéfice des programmes d'application. L'application des principes 3 et 4 conduit à l'identification de la couche présentation au dessus de la couche session.*

7. *Finalement, il ne reste plus que les processus d'application qui exécutent les traitements de l'information. Ces processus d'application et les protocoles par lesquels ils communiquent forment la couche application, la couche la plus haute de l'architecture.*

L'architecture OSI comprend donc sept couches et obéit aux principes 1 et 2 (ISO 7498, pp. 74-75).

ANNEXE D

FORMAT DE L'EN-TETE DU DATAGRAMME INTERNET

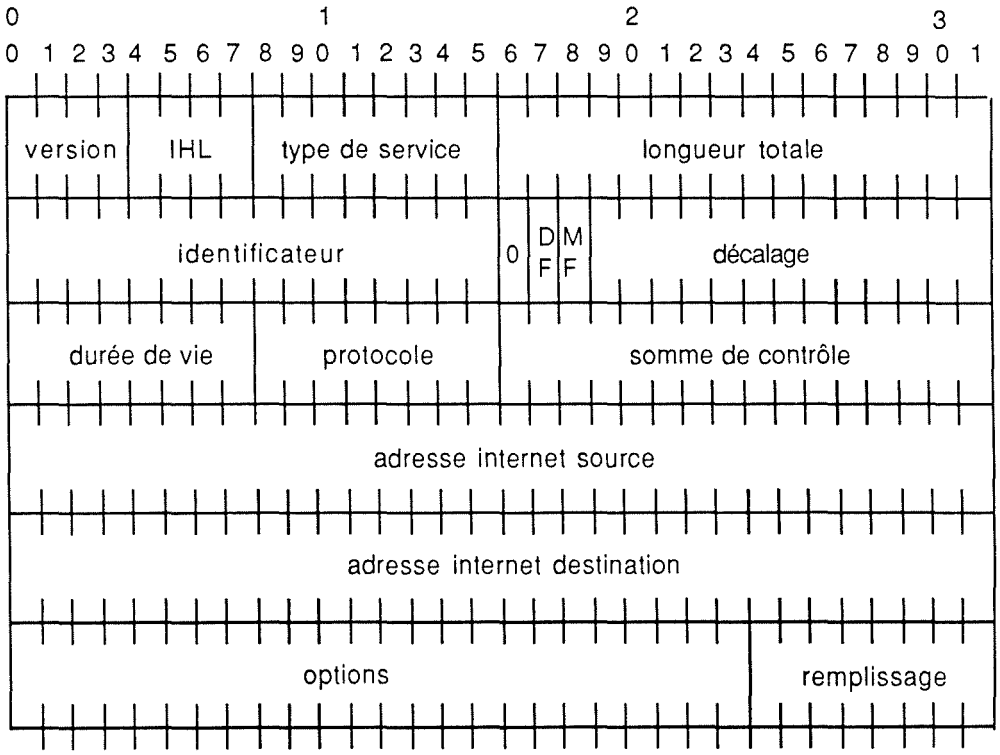


Fig. D.1. - En-tête d'un datgramme internet

ANNEXE E

FORMATS DES NPDU DE DONNEESET DE RAPPORT D'ERREUR

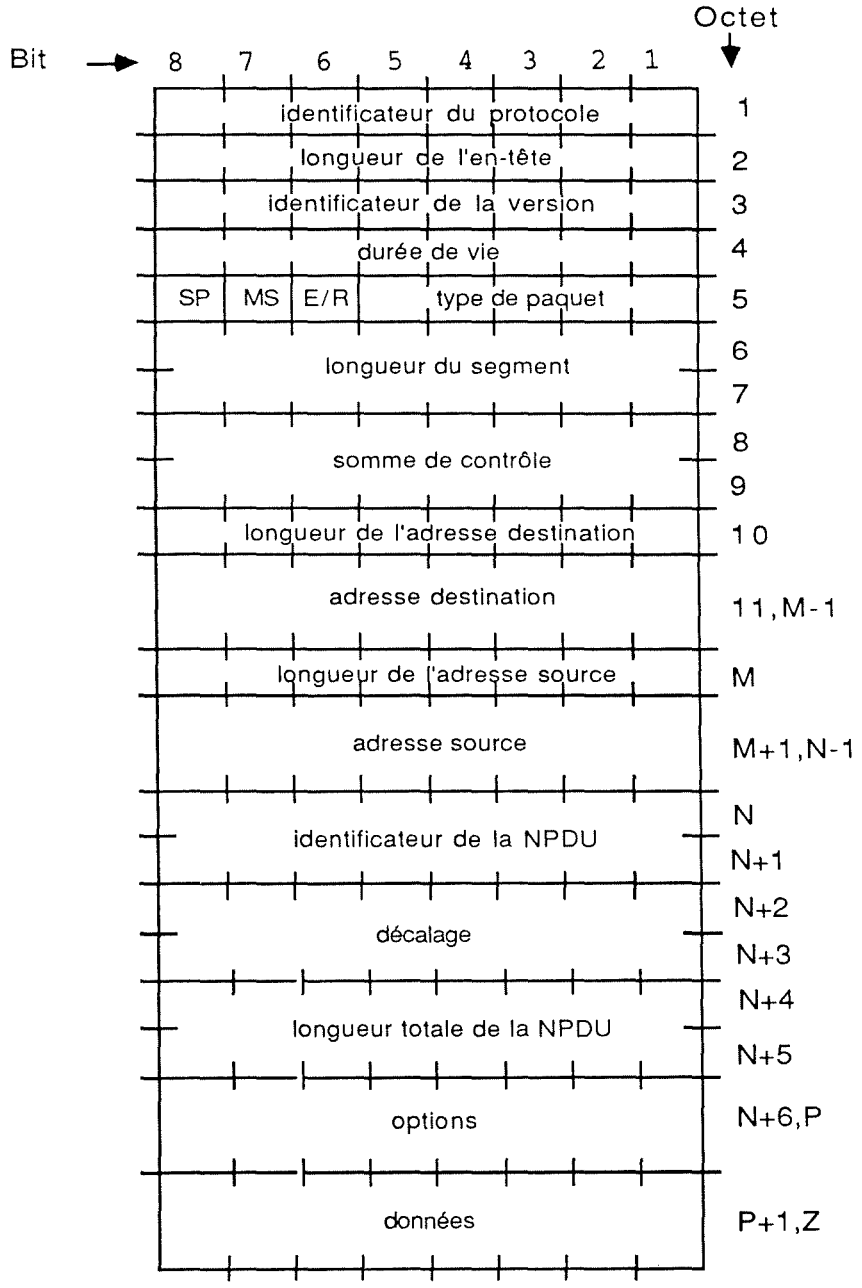


Fig. E.1. - Format des NPDU de données.

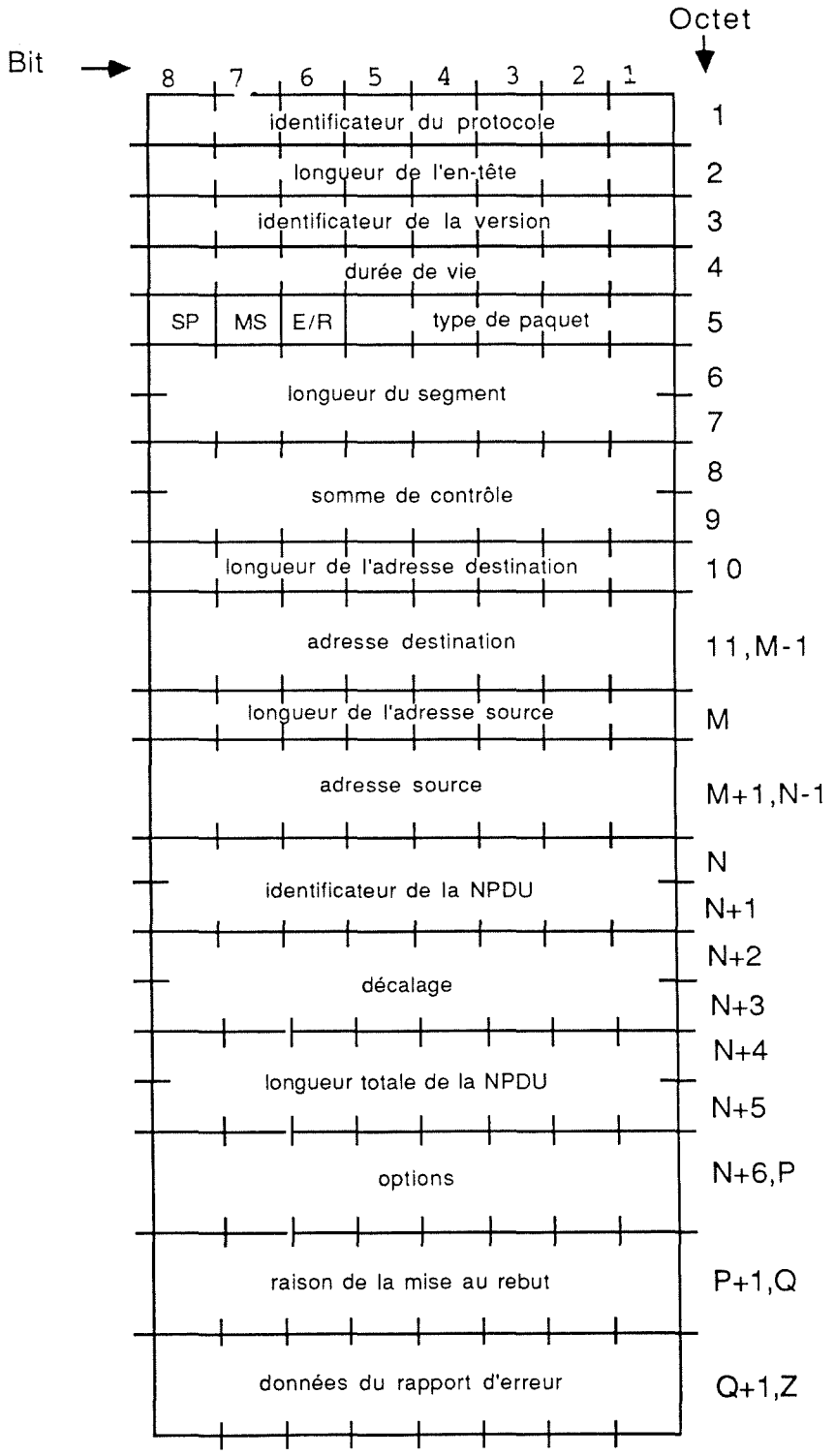


Fig. E.2. - Format des NPDU de rapport d'erreur.

ANNEXE F

FORMAT DE L'EN-TETE DU SEGMENT TCP

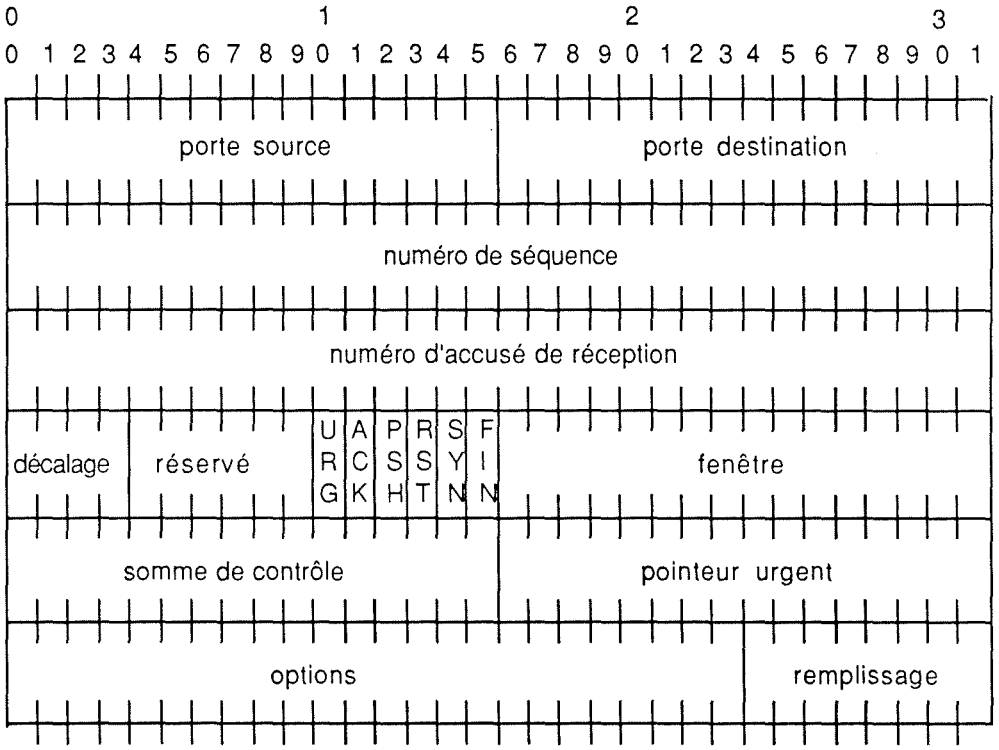


Fig. G.1. - En-tête d'un segment TCP.

Format :

- les champs *porte source* et *porte destination* (16 bits) identifient les utilisateurs occupés à se communiquer;

- le champ *numéro de séquence* (32 bits) détermine le numéro de séquence du premier octet contenu dans le champ de données du segment TCP; si le drapeau SYN est présent, ce champ détermine le numéro de séquence initial;

- le champ *numéro d'accusé de réception* (32 bits) détermine le numéro de séquence du prochain octet que le module TCP s'attend à recevoir;

- le champ *décalage* (4 bits) indique que le nombre de mots de 32 bits contenus dans l'en-tête du segment TCP; cette information est nécessaire car le champ des options est de longueur variable;

- le champ *drapeaux* (6 bits) contient 6 bits de contrôle :

    - URG est positionné à 1 si le champ pointeur urgent est utilisé; ce champ est utilisé pour l'émission de message prioritaire comme des demandes d'interruption;
    - SYN sert de l'établissement des connexions; une demande de connexion positionne SYN à 1 et ACK à 0 pour indiquer que le champ numéro d'accusé de réception n'est pas utilisé;
    - ACK permet de distinguer une demande de connexion d'une réponse positive à cette demande; l'acquittement d'une demande de connexion positionnera SYN à 1 et ACK à 1;
    - FIN permet de clôturer une connexion; ceci indique que l'émetteur n'a plus de données à envoyer;
    - PSN indique la fin d'un message;
    - RST est utilisé pour reprendre une communication devenue confuse;

- le champ *fenêtre* (16 bits) indique le nombre d'octets de données que le destinataire est prêt à recevoir à partir du numéro de séquence spécifié dans le champ numéro d'accusé de réception;

- le champ *somme de contrôle* (16 bits) est calculé sur l'en-tête et le champ de données du segment TCP;

- le champ *pointeur urgent* indique l'emplacement du premier octet de données (référencé par le numéro de séquence) situé après les données urgentes (contenues dans le champ de données du segment TCP);

- le champ *options* (taille variable) est facultatif; il peut être utilisé pour indiquer par exemple la taille maximum des segments TCP que le récepteur est prêt à accepter;

- le champ *remplissage* (taille variable) ajoute des zéros à l'en-tête de telle sorte que celle-ci se termine sur une limite de 32 bits.

BIBLIOGRAPHIE

(Bauerfeld 87)
W. Bauerfeld, "A Tutorial on Network Gateways and Interworking of LAN's and WAN's", Computer Networks and ISDN Systems, Number 13, 1987, pp. 187-193.

(Cerf 78)
V. G. Cerf, "The Catenet Model for Interworking", IEN48, Defense Advanced Research Projects Agency, July 1978.

(Cerf 83)
V. G. Cerf, E. Cain, "The DoD Internet Architecture Model", Computer Networks, Number 7, 1983, pp. 307-318.

(Ethernet 82)
Digital, Intel, Xerox, "The Ethernet, a Local Area Network, Data-Link Layer and Physical Layer Specifications", version 2.0, 1982.

(Fluckiger 87)
F. Fluckiger, "Expérience de réseaux multi-vendeurs ou l'art d'intégrer", extrait de "Les réseaux de communication", Dunod informatique,1987, pp. 15-1, 15-18.

(Hinden 83)
R. Hinden, J. Haverty, A. Sheltzer, "The DARPA Internet : Interconnecting Heterogeneous Computer Networks with Gateways", Computer, September 1983, pp. 38-48.

(Leffler 83)
S. Leffler, W. Joy, R. Fabry, "4.2BSD Networking Implementation Notes", University of California, Berkeley, July 1983.

(Macchi 87)
C. Macchi, J. F. Guilbert, "Téléinformatique", Dunod, Paris, 1987.

(Nagle 84)
J. Nagle, "Congestion Control in TCP/IP Internetworks", RFC-896, Ford Aerospace, January 1984.

(Nussbaumer 87)
H. Nussbaumer, "Téléinformatique", Presses Polytechniques Romandes, Lausanne, 1987.

(Poncelet 87)
A. Poncelet A., "Etude de logiciels de communication entre systèmes UNIX", Université Catholique de Louvain, 1987.

(Postel 81)
J. Postel, C. Sunshine, D. Cohen, "The ARPA Internet Protocol", Computer Networks, Volume 5, Number 4, July 1981, pp. 262-271.

(Pujolle 87)
G. Pujolle, D. Seret, D. Dromard, E. Horlait, "Réseaux et télématique", tome1 et tome 2, Eyrolles, Paris, 1987.

(Rose 85)
M. T. Rose, "Comments on 'Congestion Control in TCP/IP Internetworks'", ACM Computer Communications Revieuw, Volume 15, Number 4, October 1985.

(Selvaggi 83)
P. S. Selvaggi, " The Department of Defense Data Protocol Standardization Program", Computer Networks, Number 7, 1983, pp. 319-328.

(Shoch 78)
J. F. Shoch, "Internetwork Naming, Addressing and Routing", COMPCON Fall, 1978, pp. 72-79.

(Shoch 79)
J. F. Shoch, "Packet Fragmentation in inter-network Protocols", Computer Networks, Number 3, February 1979, pp. 3-8.

(Shock 81)
J. F. Shoch, D. Cohen, E. A. Taft, "Mutual Encapsulation for Internetwork Protocol", Computer Networks, Number 5, 1981, pp. 287-300.

(Stallings 85)
W. Stallings, "Data and Computer Communications", Macmillan, New York,1985.

(Tanenbaum 81)
A. Tanenbaum, "Computer Networks", Prentice Hall, 1981.

(Zhang 87)
L. Zhang, "Some Thoughts on the Packet Network Architecture", Computer Communication Revieuw, Volume 17, 1987, pp. 3-17.

(Zimmermann 83)
H. Zimmermann, J. D. Day, "The OSI Reference Model", Proceedings of the IEEE, Volume 71, Number 12, December 1983.

(RFC 768)
J. Postel, "User Datagram Protocol - DARPA Internet Program Protocol Specification", RFC 768, USC/Information Sciences Institute, August1980.

(RFC 791)
J. Postel, "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, September 1981.

(RFC 792)
J. Postel, "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC 792, USC/Information Sciences Institute, September 1981.

(RFC 793)
J. Postel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, September 1981.

(RFC 826)
D. Plummer, "An Ethernet Address Resolution Protocol", RFC 826, MIT, November 1982.

(RFC 950)
J. Postel, J. Mogul, "Internet Standard Subnetting Procedure", RFC 950, Stanford University, August 1985.

(RFC 975)
D. L. Mills, "Autonomous Confederations", RFC 975, M/A-COM Linkabit, February 1986.

(RFC 1009)
R. Braden, J. Postel, "Requirements for Internet Gateways", RFC 1009, ICI, June 1987.

(ISO/DIS 8648)
ISO, "Information Processing Systems - Data Communications - Internal Organization of the Neywork Layer", Draft International Standard ISO/DIS 8648, ISO, Geneva,1986.

(ISO/DIS 8473)
ISO, "Information Processing Systems - Data Communications - Protocol for providing the Connectionless-Mode Network Service", Draft International Standard ISO/DIS 8473, ISO, Geneva,1985.

(ISO/DIS 8348/ADD2)
ISO, "Information Processing Systems - Data Communications - Network Service Definition - Addendum 2 : Network Layer Addressing", Draft International Standard ISO/DIS 8473, ISO, Geneva,1985.

(ISO 7498)
ISO, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", Draft International Standard ISO 7498, ISO, Geneva, 1982.

(ISO/DIS 7498-3)
ISO, "Information Processing Systems - OSI Reference Model - Part 3 : Naming and Addressing", Draft International Standard ISO/DIS 7498, ISO, Geneva, 1987.

(ISO/DIS 8473/ADD1)
ISO, "Information Processing Systems - Data Communications - Protocol for providing th Connectionless-Mode Network Service - Addendum 1 : Provision of the underlying service assumed by ISO 8473", Draft International Standard ISO/DIS 8473/ADD1, ISO, Geneva, 1986.